AD-A101 860 INTEGRATED SCIENCES CORP SANTA MONICA CA F/G 15/7
CONCEPTS FOR HIGHLY INTERACTIVE AIR STRIKE PLANNING SYSTEMS (AS--ETC(U)
MAY 81 D H WALSH, G J REBANE, L R LEVI N00014-80-C-0812
UNCLASSIFIED ISC-330-2 NL

1 OF 2
AD A
101860

LEVEL

(12)

AD A101 360

CONCEPTS FOR HIGHLY INTERACTIVE AIR STRIKE PLANNING SYSTEMS (ASPS)

REPORT NO. 330-2

PREPARED FOR:

CODE 455
DIRECTOR, ENGINEERING PSYCHOLOGY PROGRAM
PSYCHOLOGICAL SCIENCES DIVISION
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

MAY 1981

INTEGRATED SCIENCES CORPORATION
Santa Monica, California

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A101

3. RECIPIENT'S CATALOG NUMBER 860

4. TITLE (and Subtitle)

CONCEPTS FOR HIGHLY INTERACTIVE AIR STRIKE PLANNING SYSTEMS

5. TYPE OF REPORT & PERIOD COVERED

TECHNICAL REPORT

6. PERFORMING ORG. REPORT NUMBER

330-2

7. AUTHOR(s)

D.H. Walsh   L.R. Levi
G.J. Rebane   R. Tash

8. CONTRACT OR GRANT NUMBER(s)

N00014-80-C-0812

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Integrated Sciences Corporation
1640 Fifth Street, Suite 204
Santa Monica, CA 90401

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

ENGINEERING PSYCHOLOGY PROGRAM, CODE 455
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

12. REPORT DATE

May 1981

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Reproduction in whole or in part is permitted for any purpose of the United States Government.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Decision Aids	Man/Computer Functional Allocation
Planning Aids	Nonlinear Programming Optimization
Naval Tactics	Command and Control
Interactive Graphics	Mission Planning

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is a report of work accomplished on one task in the project entitled Operational Decision Aids (ODA), which was initiated in 1974 by the Office of Naval Research to develop aids for Navy command and control functions and make them available for incorporation in the design of future systems. The first phase of the work produced a concept for a laboratory version of an Air Strike Planning System (ASPS) and about 2,000 lines of operational FORTRAN code. The purpose of the laboratory

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (continued)

ASPS was to experimentally test operator performances using variations of ASPS which differed in the amount of automation available to the air strike planner. Inputs to the concept formulation process about air strike planning needs and perspectives were obtained from A-7, A-6, and EA-6B pilots at Lemoore and Whidbey Naval Air Stations. The software implemented much of the user/machine interface for a computer and display system.

A concept for a fieldable version of ASPS that can be evaluated at the squadron level was developed during the second phase of the work. That concept and a recommendation for hardware to implement the concept are documented in the report.

6)

CONCEPTS FOR HIGHLY INTERACTIVE
AIR STRIKE PLANNING SYSTEMS (ASPS).

(12) 263

14) ISC
Report No. ~~330~~-2

15) Contract No. N00014-80-C-0812

9) Technical rept.,

Prepared for:

Code 455
Director, Engineering Psychology Program
Psychological Sciences Division
Office of Naval Research
Department of the Navy
Arlington, Virginia

By:

10) David H. Walsh
George J. Rebane
Lawrence R. Levi
Richard Tash

Integrated Sciences Corporation
1640 Fifth Street
Santa Monica, California 90401

11 May 1981

ACKNOWLEDGMENTS

We would like to acknowledge the support of Dr. Martin A. Tolcott, Office of Naval Research. We especially appreciate the guidance and thoughtful suggestions by Dr. Tolcott and Cdr. Kent Hull of ONR. Cdr. Hull also facilitated our access to the many sources of information about air strike planning and use of digitized terrain data that were used in our work.

SUMMARY

This is a report of work accomplished on one task in the project entitled Operational Decision Aids, which was initiated in 1974 by the Office of Naval Research to develop aids for Navy command and control functions and make them available for incorporation in the design of future systems. In previous work on the ODA program, ISC developed and tested two interactive methods of selecting (a) an air strike path through a field of ten enemy sensors and (b) aircraft speeds on each leg of the path. Both methods were implemented on a computer-driven, four-color vector graphics display. Utility for each candidate strike path was computed according to a predetermined utility function; the utility function was nonlinear and multi-modal. With one method called Operator Aided Optimization (OAO), the operator guided an optimizing nonlinear programming algorithm toward a solution. With the other method called Iterative Manual Optimization (IMO), the operator selected a candidate solution and the computer acted as a calculator by calculating and displaying the solution's utility. The operator then modified the previous solution in light of what was learned from seeing its utility. The process was repeated until the operator was satisified with the utility score achieved. An experiment using sixteen college students as subjects did not conclusively demonstrate that OAO was superior to IMO for problems having the relatively low level of complexity of the experimental problems.

The initial phase of the work reported in this document was a continuation of the previous work. ISC's aim was to develop and experimentally test OAO- and IMO-type aids that would account for most of the complexity of real-world problems by treating most of the problem factors, decision dimensions, and utility dimensions currently deemed important by air strike planners. Inputs about air strike planning needs and perspectives were obtained by visits to strike planners at Lemoore and Whidbey Island Naval Air Stations in October and November 1980 respectively. Strike planners from these Naval Air Stations also visited ISC in March 1981 and exercised on ISC's display system the operational software that had been developed up to that time. The report documents the concept and software that were developed for the laboratory version of the Air Strike Planning System (ASPS).

ODA funds for FY 81 and future years were reprogrammed in early 1981. A result of this action was that ISC received direction to devote its remaining contract effort to planning the modification and transfer of the developed technology to a fieldable desktop version of ASPS. Consequently, ISC developed during the second phase of its work a concept for a fieldable version of ASPS that can be evaluated at the squadron level. That concept is documented in this report.

During the latter stages of work, ISC concentrated on two key problem areas for a future effort to develop a fieldable planning aid for evaluation. The first of these involves the following:

1. Storing the digital terrain data that would be needed to represent real maps.

2. Development of an algorithm or modification of an existing algorithm for efficiently calculating terrain masking of hostile weapons/sensors at specified aircraft altitude levels above ground level.

ISC determined that digitized map products derived from the Defense Mapping Agency's Digital Landmass System (DLMS) Data Base are available and applicable to ASPS. The report describes several algorithms that have been developed for other programs for calculating terrain masking around a given point.

The second key problem is selecting the most cost-efficient computer and display system for implementing the system concept. ISC analyzed the capabilities of state-of-the-art graphics displays. A combination of a Megatek 7250 display and a Digital Equipment Corporation MINC 11/23 desktop computer was judged to be the best hardware available for a desktop version of ASPS.

Development of a highly interactive ASPS is recommended because ASPS will enable planners to use their time more efficiently than is possible with current manual procedures and because cost of ASPS development and purchase will be low. Low development cost is attributable to the fact that much of the design work has already been done. Low purchase cost is due to the recent arrival of exceptionally powerful, low cost desktop computers on the market.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

This report documents the results of a two-phase effort to field a desktop version of an interactive air strike planning system (ASPS) for evaluation at the squadron level. The initial phase of this work was a continuation of a multi-year participation by ISC in the Operational Decision Aids program of ONR. Effort was directed during the second phase toward a concept design that adapted a subset of the laboratory ASPS capability into the physical and computational constraints of a fieldable and relatively inexpensive desktop ASPS.

1.1 BACKGROUND

This is the sixth report by Integrated Sciences Corporation (ISC) as one of a group of contractors working on the Operational Decision Aids (ODA) program directed by the Office of Naval Research. The ODA program was initiated in 1974. Its intent is to develop a variety of decision aids and test and evaluate their usefulness to the Navy. Although the program is not tied to any specific command and control hardware system, it has focused on the functions of a Task Force Commander (TFC) and his staff. The role of ISC has been to find ways to improve user/machine communication by allocating functions between user and machine that take advantage of their respective strengths.

Under a previous contract ISC developed aids that were used to test the ability of humans to perceive complex functional relationships presented in geometric/graphical format (References 1, 2, and 3) on a computer-driven color display. The problem situation for which the aids were developed was the selection of (a) an air strike route through a field of multiple enemy sensors and (b) aircraft speeds on each leg. (Hereafter in this report, the selection of aircraft flight parameters is abbreviated to "route planning.")

One of the ISC-developed aids was called Operator Aided Optimization (OAO) using a Nonlinear Programming (NP) algorithm or OAO/NP. For the OAO/NP aid, the operator controls the algorithm by:

1. Choosing a starting point for the algorithm.

2. Stopping the optimization process of the algorithm when the utility calculated by a mathematical utility function shows diminishing returns versus time.

3. Selecting a new starting point in another region of the solution space.

Another mode of aiding the operator, Iterative Manual Optimization (IMO), was also developed. With IMO, an operator inputs a solution consisting of the route legs and speeds on each leg. The computer then calculates and displays the solution's utility and does nothing more until the next change to the route is input by the operator. Thus, with IMO, the computer acts as a calculator and all optimization is done by the operator.

An experiment was performed to determine how closely operator performance using IMO on a set of experimental problems would match performance using OAO/NP. The experiment was aimed at shedding light on the question of the importance of an optimizing algorithm as part of an aid to solving a problem having a multimodal, unsymmetric, nonlinear criterion function. The results showed that operators using IMO achieved an average score of 98.1 utility points by the end of the 15-minute experimental problems, whereas the average score was 99.4 using OAO/NP. The inference drawn from these results was that an optimizing algorithm does not greatly improve performance for problems having the level of complexity of the experimental problems. However, even for problems of this moderate complexity, use of an optimizing algorithm is worthwhile if its software cost is low and there is wide agreement among operators about the form and specific parameters of the utility function. The reason is that operators prefer using an algorithm because it makes the task simpler and less tiring.

In many situations where a planning aid will be useful, it will not be possible to pre-specify a universally acceptable form and parameters of a utility function because:

1. The importance of utility dimensions will vary from one tactical situation to another in the mind of a single decision maker.
2. Different decision makers will have different perceptions of what constitutes utility.

It is natural to ask whether optimizing algorithms have a place in aids for these situations. ISC believes they do. Our reasoning begins with the idea that, in these cases, the planning aid must be designed to do one of the following:

1. Allow the operator to select utility dimensions to be included in an overall criterion and to specify mathematical forms for combining dimensions and weights for each dimension. (It's difficult to do this well.)
2. Be free of machine calculation of an overall utility value. (The operator mentally combines the values along the utility dimensions.)

If (1) above is successfully implemented with mathematically tractable functions to the satisfaction of end users, then it is certainly reasonable to allow operator guided use of a general purpose optimizing algorithm. If (2) above is the more desirable approach, then it is reasonable to allow the operator to guide an optimizing algorithm along a single utility dimension at a time and let values along other dimensions fall where they may. In this case the operator can converge on the best solution across multiple utility dimensions through an iterative process.

1.2 CURRENT WORK

The real world of air strike planning is more complex than the problem used to compare the OAO and IMO concepts. Important real-world considerations are listed below:

Problem Factors

- capabilities and location of enemy sensors
- capabilities and locations of enemy missiles
- capabilities of enemy fighters and locations of their bases
- capabilities and locations of enemy anti-aircraft guns
- change in location of enemy defense forces during the flight time of the air strike
- terrain topography

Decision Dimensions

- x,y locations of path legs
- speed on each leg
- altitude on each leg
- when to use on-board jamming equipment
- where to direct use of electronic warfare aircraft
- weapon and fuel loading
- when and where to use standoff anti-radar weapons

Utility Dimensions

- expected damage to strike aircraft due to enemy defenses
- probability of strike mission success
- distance between target and strike aircraft when cumulative detection probability exceeds x%

The IMO and OAO aids tested in the previous work did not account for all of these dimensions. Consequently, the following question arose:

> Is the small magnitude of the performance difference between IMO and OAO observed in the study due to the relative simplicity of the problems to be solved?

This question may be put another way:

> How will the difference in operator performance using IMO- and OAO-type aids change as the dimensions of the problem, decision, and utility function increase?

Performance data from aids and experiments designed to answer this question could be important evidence supporting a decision by R&D program managers to concentrate future efforts on developing one of the aiding concepts in preference to the other.

Consequently, the original aim of the current work was to develop and experimentally test OAO- and IMO-type aids that would account for most of the real-world problem factors, decision dimensions, and utility dimensions currently deemed important by air strike planners. ISC project members visited A-7 and A-6 strike planners at Lemoore and Whidbey Island Naval Air Stations in October and November 1980 in order to gain understanding of the route planning problem as the planners see it. We also obtained inputs concerning the planners' preferences for features and characteristics of an aid. This information provided much of the basis for our initial concept for an air strike planning aid (Reference 4).

ODA funds for FY 81 and future years were reprogrammed in early 1981. The ISC air strike planning concept and software are applicable to a current fleet need; consequently, ISC received the direction to devote its remaining contract effort to planning the modification and transfer of the developed technology to a fieldable desktop Air Strike Planning System.

Our initial concept for the aid was modified as a result of additional information obtained from strike planners in February and March. A major new input was received from Marine Corps aviators at Quantico in February. They stressed the need for an aid to planning and coordinating actions in the target area. This idea had been mentioned previously at Lemoore and Whidbey, but it had not been stressed. Additional inputs were received from strike planners who visited ISC in March, reviewed our concept, and exercised on our display system the operational software that had been developed up to that time. The reviewing officers were strike planners from Lemoore and an A-6 strike planner and an EA-6B crew member from Whidbey. The major new input from these visits was the desirability of having the aid show a dynamic replay of an alternative. As before, this feature had been mentioned earlier by strike planners, but it had not been stressed.

During the latter stages of work, ISC concentrated on two key problem areas for a future effort to develop a fieldable planning aid for evaluation. The first of these involves the following:

1. Storing the digital terrain data that would be needed to represent real maps.

2. Development of an algorithm or modification of an existing algorithm for efficiently calculating terrain masking of hostile weapons and sensors at specified aircraft altitude levels above ground level.

The second key problem is selecting the most cost-efficient computer and display system for implementing the system concept.

Section 2 of this report contains our concept for an Air Strike Planning System (ASPS) that embodies most of the features recommended by the Naval and Marine Corps air strike planners consulted by ISC. Elements of this concept have been translated into operational software; these elements are noted in Section 2. Section 3 contains the results of our investigations into the two key problems summarized in the previous paragraph. Our recommendation for development of a highly interactive Air Strike Planning System (ASPS) is in Section 4. Documentation of software developed during this project is presented in the appendices.

2.0 CONCEPTS FOR AN AIR STRIKE PLANNING SYSTEM (ASPS)

A concept and software for a laboratory version of an air strike planning system were developed during the initial phase of the contract. The laboratory version was to be used to test hypotheses about the performance of planners on problems that were virtually as complex as real planning problems. The most important results of the tests were expected to be the comparisons between performance using iterative manual optimization and operator aided optimization. The concept for the laboratory version of ASPS is documented in Subsection 2.1.

A fieldable ASPS for evaluation by an operational squadron requires capabilities that are not needed for the laboratory tests. The most important such capability is representation of actual terrain on the system's computer-driven display. Subsection 2.2 describes:

1. Capabilities that would be different in the fieldable version from the laboratory system described in Subsection 2.1.

2. Capabilities that strike planners desire but that are not needed to test hypotheses about OAO and IMO in a laboratory experiment.

2.1 CONCEPT FOR A LABORATORY VERSION OF ASPS

Elements of the concept which have been implemented in operational software at ISC are indicated by an asterisk within parentheses preceding the applicable concept elements, i.e., (*). The software structure and data file structure for these concept elements are discussed in the appendices. The data file structure for the completed ASPS system will be enhanced to allow for additional capabilities, but we expect its basic structure to remain the same.

The hardware for the laboratory version of ASPS includes a Sperry Univac V-73 minicomputer, a four-color vector graphics display, a 32-key special function keyboard, and a trackball. The special function keys and trackball are the instruments which allow the planner to communicate with the computer. Their specific uses are explained in detail below.

2.1.1 Problem Setup and Controls

(*) 2.1.1.1 Terrain Modeling; Modeling and Plotting Enemy Defenses. Scenario terrains will be developed off-line and stored in the computer. Terrain features will include:

1. Four Above-Ground-Level (AGL) values of altitudes
2. Cities
3. Forests
4. Lakes
5. Roads

(*) Locations of radar, missile, and anti-aircraft (AA) gun sites will also be stored with terrain. Contours of what can be seen for each AGL on a line-of-sight basis from each radar, gun, and missile site will be stored and displayed on operator request. The composite contours of what can be seen for each of the four AGL's will also be stored for the group of radars, the group of missile sites, and the group of AA gun sites.

(*) Each element of terrain and displayed information about defenses will have a display "priority" code attached to it. These are:

PRIORITY I	PRIORITY II	PRIORITY III
Terrain features	Visually significant points	Detectability contours for radars
Radar sites	Radar significant points	Reachability contours for missiles and AA weapons
Missile sites	(These are inputs by the operator. See Section 2.1.2 on Variable Background)	

Priority I information will always be displayed. Priority II and Priority III information will be displayed by operator command.

(*) 2.1.1.2 Beginning Operations. At the beginning of system operation, the following menu appears:

"Select MAP 1
2
3
4"

The operator indexes a cursor to the desired map and then pushes ACCEPT.[1] The screen shows Priority I information, namely, the radar and missile sites, and the physical terrain.

(*) After selecting the map from the menu, the operator uses the number pad to input the plan number.[2] (See Figure 1.) A message indicating the number of alternative plans already stored appears below the screen. For each map, Plan 1 is always the basic map without significant points or path. (If only the map exists without any plans, then the user will not be asked for a plan number but, instead, will be placed in the Director Mode). After inputting and accepting the plan number, the path and significant points will be displayed. The user is now in the Director Mode.

(*) 2.1.1.3 Director Mode. The Director Mode controls the flow of the program as specified by the operator. The active special function keys are lighted and await a user response. The key pressed specifies the next activity to be worked upon by the operator.

(*) Throughout all procedures, the latitude and longitude position of the cursor will be displayed whenever the cursor appears on the screen. Only those keys which are lighted will be active at any time during the flow of the program.

---

[1]The number of maps available is limited only by computer memory.

[2]A plan is defined to be a path, speed, and altitude for each aircraft type in the strike plan. The plan for strikes including EA-6B aircraft includes the points in the path where electronic warfare equipment is turned on.

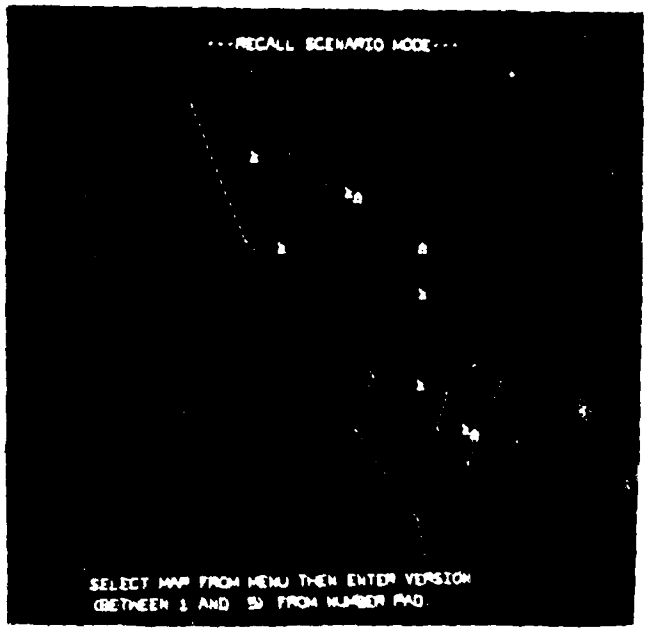


Figure 7. A Map Recalled by the Planner

2.1.1.4 Initialization. The user is given the opportunity to initialize certain constraints and inputs in the program such as fuel allocation, amount of fuel at recovery, and local time at beginning of the mission. There is an INIT key if he wishes to make changes; otherwise, default values will be used.

(*) 2.1.1.5 Screen Layout, Symbology, and Special Function Keys. Figure 2 shows the preliminary screen layout for the system. The display screen is laid out with the map in the center and miscellaneous information around the sides. The miscellaneous information includes menus, task and error messages, latitude and longitude position of the cursor, the current mode, a table of leg speeds and AGL's along the path, and other pertinent information. Table 1 describes the symbol and line conventions used in the ASPS program. Figure 3 shows the layout of the special function keyboard.

2.1.2 Variable Background

(*) The user is given the capacity to add elements which he considers important to the map. There are two types of these, namely, radar significant points (RSP) and visual significant points (VSP). The RSP's are points detectable by radar and VSP's are points that can be seen visually. These types of points might be relevant when selecting the path to and from the target area.

(*) When the operator presses SIGNIFICANT POINTS key, a cursor appears on the screen and is maneuvered with the trackball. All radar significant points are located, followed by visually significant points. The operator moves the cursor to the desired location and presses the ACCEPT key. An "R0" appears at that position. He moves the cursor to the location for the next radar significant point and ACCEPTs, which displays an "R1". This process continues up to a maximum of ten radar significant points. If the RETURN key is pressed before ten radar significant points are located, the user will then be asked to locate the visually significant points.

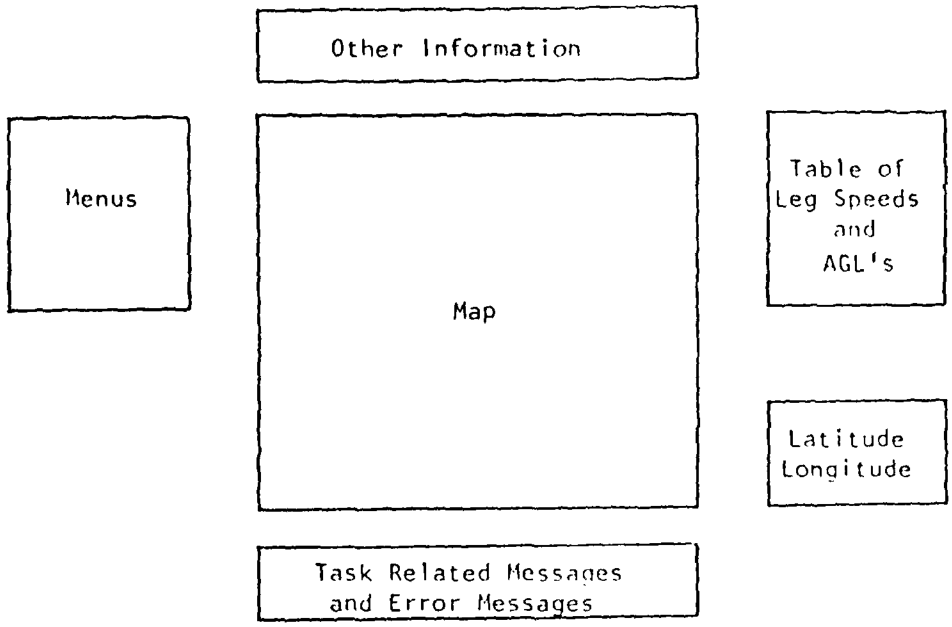
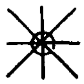


Figure 2. Preliminary Screen Layout for the Air Strike Planning System (ASPS).

Table 1. TABLE OF SYMBOLS

Symbol/Line Type	Description	Color
	Launch Point	Red
	Target point	Red
	Recovery Point	Red
○	Way Point	Green
	SAM Site	Orange
	Sensor Site	Orange
Ri $i = 0,\ldots,9$	Radar Significant Point	Yellow
Vi $i = 0,\ldots,9$	Visually Significant Point	Yellow
⊕	Cursor	Yellow
Solid Line	Path	Green
Dash-Dot Line	Line Connecting Range and Bearing Reference Point to Cursor	Red
Dashed Line	Sensor Contours	Red
Dash-Dot Line	Shoreline	Green
Dotted Line	Hills	Yellow
Dash-Dot Line	City	Yellow

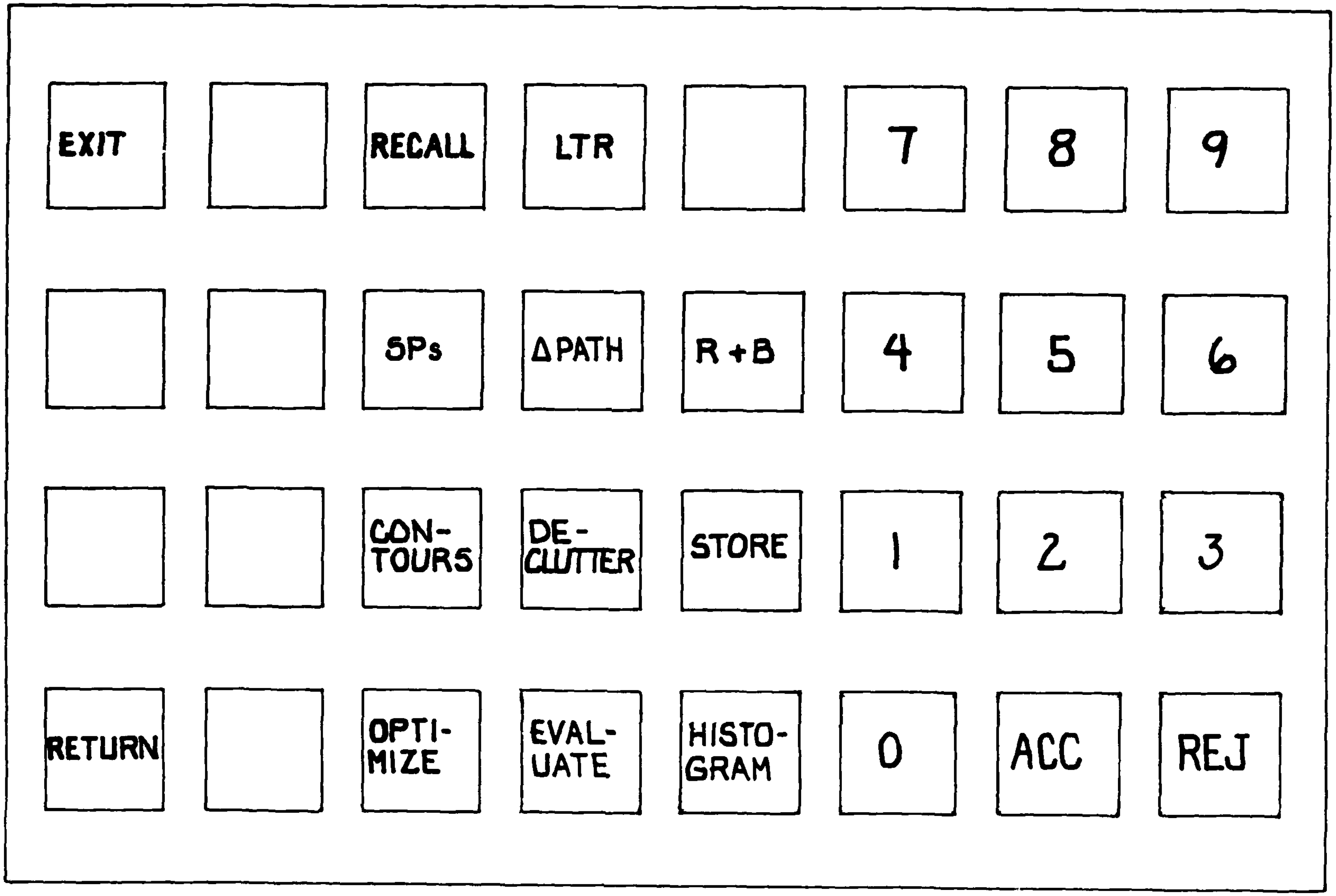


Figure 3. Function Key Setup.

(*) The process for visually significant points is the same as for radar significant points. A "V0", "V1", etc. appears on the screen identifying the locations of the visually significant points. Pressing the RETURN key will return control to the Director Mode.

2.1.3 Path Manipulation - IMO

This section describes how to change the path and path features. The basic idea is to find the path which yields the best value of the figure of merit (FOM) and satisfies the fuel constraint. The system enables the user to alter the route, speed, and AGL along the route until an acceptable FOM is determined. The FOM is calculated by use of the EVALUATE key.

(*) 2.1.3.1 Key "LTR". There is a key "LTR", which stands for Launch Point, Target Point, Recovery Point. When this key is pressed, a message appears below the map:

LOCATE LAUNCH POSITION WITH CURSOR.
PRESS 'ACCEPT' KEY WHEN CURSOR IS OVER DESIRED POSITION.

A cursor is moved to the desired launch point and accepted with the ACCEPT key. A symbol "L" will appear for the launch point.

(*) Similarly, this procedure is followed in turn for the target and recovery nodes with the appropriate change in the message. A "T" appears to identify the target point and an "R" for the recovery point.

(*) After locating the recovery point, a line connecting the route from launch to target to recovery is displayed. A Leg/Speed/AGL Table appears, indicating the speed and altitude for each respective leg. The default values are 360 knots at 500 feet.

(*) After the recovery position is identified, a menu appears that enables the user to specify the type and quantity of stores carried by the aircraft. The menu has the following options:

"• 500 lb Bomb
• 1000 lb Bomb
• Missile
• Fuel tanks"

The user ACCEPTs he desired type of store from the menu and then uses the number pad to describe the quantity to be carried on the mission. The number selection then appears to the right of the menu item. When the user presses the RETURN key, the numbers of each desired store type are entered in the master data file. If the user wishes to change the quantity of a previously selected type, then the user indexes to the desired type and enters the desired quantity on the number pad. This overrides the previously stored number.

(*) 2.1.3.2 Key "CHANGE PATH". The CHANGE PATH key allows the user to select a waypoint that he intends to alter. The following message appears:

-- SELECT NODE --

USE TRACK BALL TO POSITION THE CURSOR OVER THE
DESIRED NODE AND PRESS THE 'ACCEPT' KEY.

Figure 4 shows a path which has been manipulated using the CHANGE PATH mode.

(*) A cursor which is moved by the trackball appears in the scenario. When the cursor is moved over one of the nodes, the node blinks and the ACCEPT key is activated. Pressing the ACCEPT key displays the menu:

"• Add Node
• Delete Node
• Move Node
• Change Leg Speed
• Change Leg AGL
• Change Stores"

A menu cursor moves to the next option in the list each time the REJECT key is pressed. When the cursor is at the option desired, pressing the ACCEPT key will select it. The only option available for the recovery node is "Move Node." The CHANGE PATH key is active only if the launch, target, and recovery points exist.

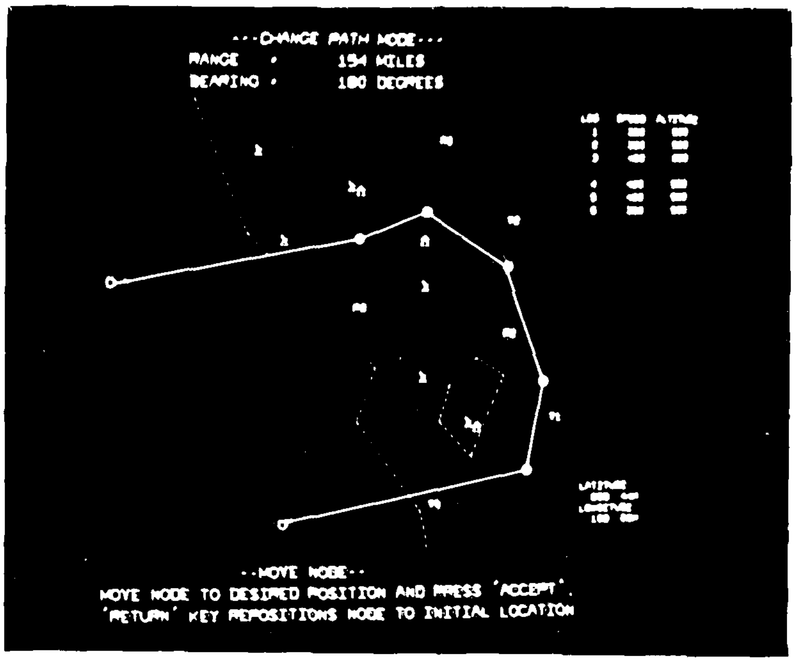


Figure 4. A Path Which Has Been Manipulated Using Change Path Mode

(*) <u>Add Node</u>

When this option is selected, a new waypoint appears halfway between the current node and the next along the flight path. Also, a position in the Leg/Speed/AGL Table appears for the leg following the new waypoint. (See Table 2 for an illustration.) In general, this is used in conjunction with Move Node to position the new node. After a node has been added, the user returns to CHANGE PATH.

(*) Deleting the last selected node can be accomplished by ACCEPTing the Delete Node option from the menu. This causes the node to be deleted, the path connected to its neighboring nodes, and the corresponding leg eliminated from the Leg/Speed/AGL Table. (See Table 3 for an illustration.) The Launch, Target, and Recovery points cannot be deleted. Immediately after Delete Node is ACCEPTed, the user returns to CHANGE PATH mode.

Table 3. Illustration of What Happens When a Waypoint is Deleted.

OLD TABLE

(Node 3 is to be deleted.)

Leg	Spanning Waypoints	Speed	AGL
1	$N_1 - N_2$	480	500
2	$N_2 - N_3$	300	200
3	$N_3 - N_4$	360	100
4	$N_4 - N_5$	420	500

NEW TABLE

Leg	Spanning Waypoints	Speed	AGL
1	$N_1 - N_2$	480	500
2	$N_2 - N_4$	300	200
3	$N_4 - N_5$	420	500

(*) <u>Move Node</u>

When this item is accepted from the menu, the node to be moved remains blinking and a message below the scenario appears:

-- MOVE NODE --

MOVE NODE TO DESIRED POSITION AND PRESS 'ACCEPT.'
'RETURN' KEY REPOSITIONS NODE TO INITIAL LOCATION.

Table 2. Illustration of What Happens When a Waypoint is Added.

OLD TABLE (A new node is to be added between nodes 3 & 4.)

Leg	Spanning Waypoints	Speed	AGL
1	$N_1 - N_2$	360	1000
2	$N_2 - N_3$	420	500
3	$N_3 - N_4$	480	200
4	$N_4 - N_5$	540	1000

NEW TABLE

Leg	Spanning Waypoints	Speed	AGL
1	$N_1 - N_2$	360	1000
2	$N_2 - N_3$	360	1000
3	$N_3 - N_{3.5}$	480	200
4	$N_{3.5} - N_4$	480	200
5	$N_4 - N_5$	540	1000

(*) If an attempt to add a node too close to neighboring nodes is made, an error message appears:

-- ERROR --

WAYPOINT NOT ADDED SINCE IT LIES TOO CLOSE TO
SURROUNDING WAYPOINTS!
PRESS 'RETURN' KEY TO CONTINUE...

(*) If an attempt is made to add more than ten way points, the following error message appears:

-- ERROR --

THE MAXIMUM ALLOWABLE WAY POINTS -10- HAS
BEEN EXCEEDED.
PRESS 'RETURN' KEY TO CONTINUE...

If either of these messages appear, pressing RETURN automatically puts the user into CHANGE PATH mode.

The node can be moved around the map with the trackball. Pressing the ACCEPT key fixes its location and returns the user to the CHANGE PATH mode. If the user moves the node but wishes to return it to its original position, then the RETURN key should be pressed. The node will return to its starting position and the user will be in the CHANGE PATH mode.

(*) Change Leg Speed

When this option is ACCEPTed, the leg, preceding node, and corresponding entry in the Leg/Speed/AGL Table will blink and the following menu appears:[1]

"• SPEED I = 300 KNOTS
• II = 360
• III = 420
• IV = 480
• V = 540"

When the desired speed for that leg is ACCEPTed, the Leg/Speed/AGL Table will reflect the change in its corresponding entry, and the user is returned to the CHANGE PATH mode. If no change is desired, the RETURN key puts the user into the CHANGE PATH mode.

(*) Change Leg Altitude

This procedure is very similar to Change Leg Speed. The following menu appears:

"• AGL I = 250 FEET
• II = 500
• III = 750
• IV = 1000"

[1] Pilots almost exclusively use integer multiples of 60 knots because time and distance calculations are simple to make. For example, 480 knots corresponds to eight nautical miles per minute. If a leg is 36 miles long, the time between turning points is 4 1/2 minutes. During the mission the pilot can therefore use his watch as an aid to determining when to turn. Thus, integer multiples of 60 knots are the only planning choices needed for the aid.

When an item is ACCEPTed, the corresponding altitude entry in the Leg/Speed/AGL Table is changed and the user returns to the CHANGE PATH mode. Pressing the RETURN key leaves the altitude unchanged and returns the user to the CHANGE PATH mode.

2.1.3.3 Inputs for Electronic Warfare (EW) Aircraft. Key "EW A/C" brings up the following menu:

"ROUTE

- Same Route as Attack a/c
- Different Route"

If the operator ACCEPTs "Same Route," then the program will check that speed constraints are not violated and store the same route for the EW aircraft as already planned for attack aircraft. If operator ACCEPTs "Different Path," then the operator defines the EW aircraft route in the same way as was done for the strike aircraft. At any point along the route the operator may activate or deactivate electronic warfare equipment aboard the EW aircraft. The program calculates the degradation effect of this equipment on each enemy defense system and the joint effect on overlapping systems.

2.1.3.4 Key "EVALUATE". There is an EVALUATE key. The user selects with the cursor, the beginning and ending node for which a Figure of Merit (FOM) will be calculated. The computer then calculates the value of the FOM for the user's Iterative Manual Optimization (IMO) solution and displays the local time of arrival at each waypoint, the fuel consumed for each leg, and the FOM for each leg.

2.1.4 Operator Aided Optimization (OAO)

2.1.4.1 Overview of OAO. Instead of using the IMO trial-and-error method of finding the best path, speed, and AGL to the target, the operator can have the computer use dynamic programming procedures to calculate the optimal route. First, the user defines a corridor boundary within which the optimization is to occur.

Next, a grid whose size is determined by the operator, is superimposed over the region. Finally, constraints on the speed and AGLs over the region being optimized are input by the operator. The optimal route between grid points for the given constraints is calculated and displayed by the computer. The operator can then modify the route (with the CHANGE PATH key) to make it practical, yet follow the guidelines set by the optimal path. This user/machine interaction has led to the name "operator aided optimization." The operator can use the ΔFOM key to ask the computer to define a region within which the routes are less than optimal, but still within an acceptable percentage of the optimal FOM.

2.1.4.2 Key "OPTIMIZE". The following, (a) - (d), happen in sequence:

(a) "Corridor Boundary"

The user first must select an entry and exit node on an already-defined route. A closed contour is drawn using the trackball and ACCEPT key. The entry and exit nodes exist along the boundary. When the contour is finished, the message, "Accept or Reject Corridor," appears. If the operator presses REJECT, the boundary is erased from the screen and the user must define a new corridor. If the ACCEPT key is pressed, then the corridor is set and the user then proceeds to the next step.

(b) "Grid Size"

The operator must now select a grid size from the following menu:

"• Coarse
• Medium
• Fine"

The coarse size yields a solution which is obtained more quickly but is less precisely optimal than the others. Now the user must specify the speed constraints.

(c) "Speeds"

The following menu appears:

- "300
- 360
- 420
- 480
- 540"

All speeds ACCEPTed are considered by the dynamic programming algorithm when it is calculating the optimal solution, i.e., different legs may have different speeds, chosen from the list of ACCEPTed speeds.

(d) "AGL's"

A menu of AGL's now is displayed. All AGL's ACCEPTed by the operator are considered by the dynamic programming algorithm when it is calculating the optimal solution.

The computer now calculates and displays the optimal plan between the corridor entry and exit points for the set of constraints (corridor boundary, grid size, speeds, and AGL's). The display shows the route, AGL for each leg, speed for each leg, and the cumulative values of the Figure of Merit (FOM) at the end of each leg. The "optimal" plan will usually not be one that is practically navigable from a strike leader's point of view. Therefore, the planner must now input a modified plan. This is done by entering the CHANGE PATH mode and inputting a feasible approximation to the "optimal" plan.

2.1.4.3 "ΔFOM". Pressing this key brings up the prompt, "Use number pad to input acceptable degradation percentage in FOM." When the user has input the desired percentage, the computer calculates and draws the boundary around the region that contains solutions whose FOM values lie between (a) optimal and (b) optimal - ΔFOM.

2.1.5 Operator Aids

The ASPS program provides several operator aids to be used when needed. They don't affect the analysis but will be useful as aids for planning the airstrike. These aids can be categorized into two groups: modal and non-modal.

(*) The non-modal aids are the R&B key and DECLUTTER key. The R&B key allows the user to identify the distance and angle of the cursor from a reference point. The DECLUTTER key allows the user to blank out contours and/or significant points from the screen.

The modal aids are the Histogram Mode and Contours Mode. The Histogram Mode shows a histogram of the fuel consumed and FOM for each leg along the flight path. The Contours Mode allows the user to display contours that show for a desired AGL the outer limits of defense capability for hostile sensors, surface to air missiles (SAM's), and anti-aircraft (AA) guns.

(*) 2.1.5.1 Key "R&B". The RANGE/BEARING key is active whenever the operator is in the mode that uses the cursor. The following message appears:

Above the scenario:

RANGE = MILES
BEARING = DEGREES

Below the scenario:

-- RANGE AND BEARING --

SELECT REFERENCE POINT WITH CURSOR
PRESS 'ACCEPT' KEY TO FIX REFERENCE POINT
PRESS 'RETURN' KEY TO EXIT.

The cursor is moved with the trackball to the desired reference point and the ACCEPT key is pressed.

(*) As the cursor is moved, a line is attached from the reference point to the cursor. The range (in nautical miles) and bearing (from north) are measured from the cursor to the reference point and are continually updated and displayed.

This process continues as the user proceeds in the mode from which RANGE/BEARING was called. The range and bearing are turned off when the cursor position is ACCEPTed or the mode has been exited. If the user is in the CHANGE PATH mode and has selected the "Move Node" option, the displayed range and bearing figures are calculated from the moving node instead of the cursor. (See Figure 5.)

(*) 2.1.5.2 Key "DECLUTTER". There is a "DECLUTTER" key. This key allows the user to display selected contours and significant points. The "Declutter Key" works on a cycle of four presses. Each press will, in turn, perform the following functions:

1. Shows Terrain, Significant Points and Contours
2. Shows Terrain and Significant Points
3. Shows Terrain and Contours
4. Shows Terrain Only.

2.1.5.3 Key "HISTOGRAM". After the computer has calculated the value of the FOM for a plan, the user may choose the histogram option by pressing the key with this name. The histogram will graphically convey to the user the value of the FOM and the fuel consumption for each leg.

2.1.5.4 Key "CONTOURS". This key enables the operator to see displays of defense capability contours. The options are given on the following menu:

"Select Contour Option

- Sensors
- SAM/AA
- Both
- Composite
- Composite, SAM, AA"

After selecting the option, the operator uses the number pad to select an AGL between 1 and 4. Figure 6 shows contours for both SAM and sensor sites. If no contour exists for a particular option at a certain AGL, the program returns to the Director Mode.

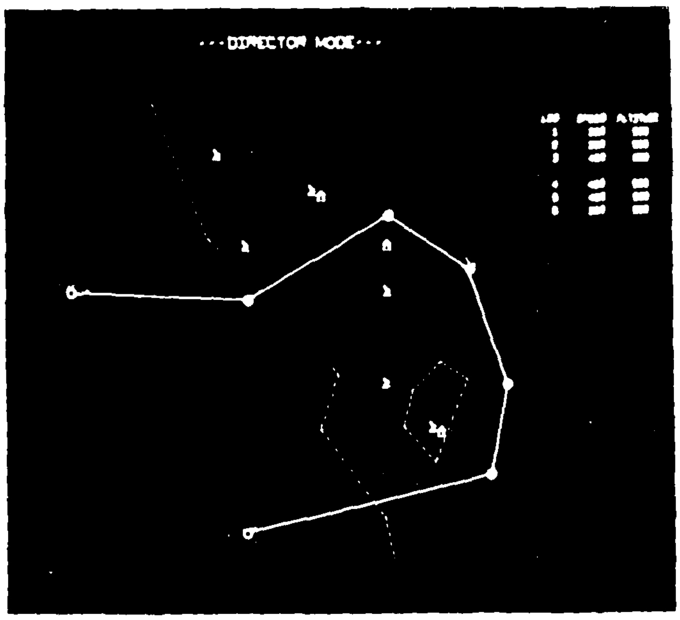


Figure 5 Illustration of Range and Bearing Use During "Change Path" Operations

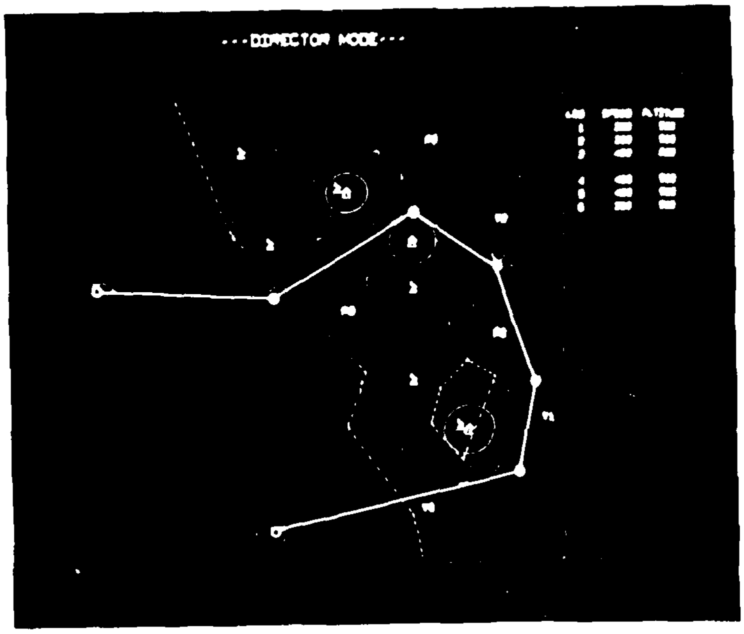


Figure 6. Defense Capability Contours for a Particular AGL

Sensors Option - This allows the user to select individual sensors to display or turn off their contours, or to turn on/off all the contours.

SAM/AA Option - The user sees all threat level contours for each of the SAM/AA sites at the given AGL.

Both - This is a combination of the above two options.

Composite - This shows the composite contours for all the sensors.

Composite, SAM, AA - This option displays the composite along with the SAM and AA contours.

2.1.6 Data File Communication

(*) The following keys are the only keys which interact with data files stored on disk. The STORE key takes the current plan and puts in onto disk, the RECALL key brings up a previously constructed plan from disk, and the EXIT key updates the Admin data file (see Appendix B).

(*) 2.1.6.1 Key "STORE". Pressing this key allows the user to save work done. When this key is pressed, a message appears below the screen indicating the available plans and the current plan number. "Plan" 1 is restricted to be the map only and is not available for storage. As many as five plans for each map can be saved. Caution: If a plan is stored in a plan number which is different from the one where it was recalled, then information previously residing in the second plan location will be lost. For example, if a plan is recalled from plan location #2, revised, and stored in plan location #4, then information previously stored in plan location #4 will be lost.

(*) 2.1.6.2 Key "RECALL". There is a RECALL PLAN key. When this key is pressed, the following message appears:

LAST PLAN IS NUMBER (e.g., 4)
BEST PLAN IS NUMBER (e.g., 3)
SELECT PLAN (1-N) USING NUMBER PAD

N is the number of plans available for recall. The operator uses the number pad to select the plan number he wants to see. The procedure is similar to that described in Subsection 2.1.1, Beginning Operations. If the RETURN key is pressed, the previous plan is displayed.

(*) 2.1.6.3 Key "EXIT". The EXIT key is used at the conclusion of the planning session. It serves two purposes:

1. To allow for normal termination of the ASPS program
2. To update the "Admin" data file.

It is important to use this key when finished, otherwise the next planning session may not properly reflect the current session's work.

2.1.7 System Calculation Capability

The system will have the following calculation capabilities:

1. Fuel consumption for a particular aircraft will be calculated as a function of speed, altitude and drag count for each leg of a route. (The model for this is obtainable from COMLATWING at Lemoore NAS.)
2. System will calculate time of arrival at each waypoint in terms of local time and time from launch.
3. The Figure of Merit (FOM) will be calculated for each user-input (IMO) solution. This will include times of arrival of waypoints, FOM for entire route, and elements of FOM (e.g., probability of survival) for each leg of a route.
4. System will compute the joint capability of enemy search radars at any point on the terrain.[1]
5. System will compute the joint capability of enemy missile sites at any point on the terrain.[1]
6. System will compute the joint capability of enemy AA guns at any point on the terrain.[1]
7. System will compute degradations to radar and missile performance as a function of EW aircraft position over time of mission.

---

[1]The dimensions of capability for each type of defense site are a matter to be decided in consultation with end users.

8. System will compute the dynamic programming solution between a user-designated pair of points after the user has input the constraints.

9. System will use the dynamic programming algorithm to compute and display the corridor containing solutions ΔFOM poorer than the "optimal" solutions.

10. System will store the four most recent solutions and the best solution to date.

2.2 CONCEPT FOR A FIELDABLE VERSION OF ASPS

As noted earlier, this subsection describes:

1. Capabilities that would be different in the fieldable version from the corresponding capabilities in the laboratory version described in Subsection 2.1. Consequently, descriptions of capabilities which would be the same in the fieldable version as in the laboratory version are not repeated here.

2. Capabilities that strike planners desire but that are not included in the laboratorv version because they are not needed to test hypotheses about OAO and IMO in a laboratory experiment.

2.2.1 Problem Setup

2.2.1.1 Type of Scenario. At the beginning of a session, the planner will see a prompt on the display:

"Type of Scenario
- Land Target
- Sea Target"

If a land target is ACCEPTed via the function keyboard, then the planner proceeds to select the desired map and then performs the initialization step covered in Subsection 2.2.1.2.

If a sea target is ACCEPTed, the user must then establish an enemy order of battle. This is done by selecting an enemy ship type from a menu and then positioning the symbol for that type on the display with the display peripheral, e.g., trackball or joystick. The latitude/longitude readout of the symbol at the lower right of the display will enable the planner to position the ship. This process is repeated for each ship in the enemy formation or disposition.

The planner may now define single-ship movements or the movement of a multiple-ship formation. To define a single-ship movement, the planner locates nodes of the route with the trackball or joystick. When the route is completely drawn, the arrival time at each node is selected by inputting a speed of advance in knots on the number pad. To define a formation movement, the planner first designates the members of the formation with the trackball or joystick and then designates the formation guide. The formation route is drawn from the location of the formation guide in the same manner as for a single-ship movement and foramtion speed is also specified the same way. Routes parallel to the guide ship are automatically defined for the other ships in the formation.

2.2.1.2 Initialization. The locations of enemy airfields are added to the terrain data in the appropriate map file. Types of interceptors at each airfield are stored and associated with the airfield's location. All other elements of initialization remain the same as defined for the laboratory version of ASPS.

2.2.1.3 Digitized Maps. The Defense Mapping Agency (DMA) has digitized map products (Reference 5). Off-line processing of this digitized data would be done using an already-available algorithm to calculate and store topographic contours and coastline for a region of interest. Thus, when a planner requests a particular map and scale, the display shows the topographic contours for the region. (Subsection 3.1 has additional information about DMA map data and their algorithm that calculates topographic contours.)

2.2.1.4 Display of Terrain Features and Defense Capability Contours. The contours for roads, rivers, lakes, and cities in a region of interest will be drawn off-line prior to a planning session and the points defining those contours will be superimposed on the topographic contour data. Radar significant points (RSP's) and Visually Significant Points (VSP's) with user-defined labels such as $\left\{\begin{array}{c}\text{VSP3}\\ \text{Tower 1}\end{array}\right\}$ will be designated either off-line by an operations or intelligence staff member or on-line by the planner. They will be superimposed on the other terrain data.

Calculations of each site's defense capability contours[1] will be based on the site's basic range envelope and the effects of terrain masking around the site. The calculation will be done on-line providing that the execution time does not cause a significant delay for the planner. However, calculation and display of joint capability contours for multiple sites will be done on-line. (Several algorithms for calculating terrain masking from a particular point using DMA data are available. See Subsection 3.1 for additional information.)

2.2.2 Variable Background

The variable background elements for the laboratory version of ASPS are:

- Visually significant points (Declutter priority II)
- Radar significant points (Declutter priority II)
- Detectability contours for radars (Declutter priority III)
- Reachability contours for missiles and AA weapons (Declutter priority III)

For land targets, the fieldable version of ASPS will have a display of a reachability circle around each enemy airfield that corresponds to the combat radius of the interceptor type designated at that airfield. For the war at sea scenario, the display will show the detectability contours for seaborne radars as a function of altitude and the reachability contours for seaborne guns and missiles. Both displays will have declutter priority III.

2.2.3 Route Planning

The first step in the route planning process will be designation of aircraft type used on the route. The alternatives available in a production version of ASPS would probably include all tactical aircraft in the Navy's inventory. A

[1]The dimensions of capability for each type of defense site are a matter to be decided in consultation with end users.

version for field test and evaluation would probably include a subset of these, such as A-6, A-7, and EA-6B. Depending on the type selected, the user will next designate the mission, e.g., attack or tanker. (The mission is automatically electronic warfare when EA-6B is selected.) When the tanking mission is selected, the tanker aircraft automatically receives a standard fuel load and weapon stores (if any) for that mission.

If the attack mission is designated, then the next step is selection of stores. Again, a production version of ASPS would probably include all stores available for the aircraft types handled by the aid. A version for field test and evaluation would probably have a subset of these, such as 500- and 1000-pound bombs, an air-to-surface missile, external fuel tanks, and an anti-radar missile (ARM's). All of the above steps are performed using the function keyboard and displayed menus in the same manner as described in Subsection 2.1.

The planner will then plan the route of the designated aircraft type and mission using the basic procedures for iteractive manual optimization (IMO) and/or operator aided optimization (OAO) as described in Subsection 2.1. Additional features in the fieldable version of ASPS are:

1. The planner may designate a point along the route where an anti-radar missile is fired.

2. The planner may designate a point where a refueling takes place. The computer transfers an amount of fuel to the attack aircraft which is lesser of (a) fuel needed to top off the attack aircraft or (b) fuel aboard tanker aircraft less fuel needed by tanker aircraft to return to carrier.

At any time after a complete route is established, the planner may elect to see a dynamic (movie-like) replay of the flight over mission time. This will include the flight of ARM's as well as aircraft. The planner may store a plan for one aircraft type and begin to plan a route for another aircraft that overlaps (in time) the mission of the previously planned route(s). The line segments displaying the geographic route of a previous plan remain on the screen while the next plan is being devised. When the planner elects the dynamic replay option after the n^{th} mission has been planned, there will be an option to see the dynamic replay of only the n^{th} mission or the simultaneous replay of all missions planned for this map area thus far.

The planner will also have the option to designate a specific defense site which is to be deleted from the enemy's joint capability. A revised Figure of Merit (FOM) for the plan can then be calculated and compared to the FOM when the enemy has full capability. This feature will help in assessing need or desirability of making a particular site the object of an attack.

2.2.4 Air Strike Target Area Planning (ASTAP)

The basic elements of the ASTAP mode of ASPS are shown in Figure 7. The air strike planner brings to the problem the types of data indicated. These data are mission-specific and must be input to the system to begin the planning session. The synoptic data relevant to strike planning in general are already contained in the system bulk memory. It is accessed as needed by the specific planner-supplied mission parameters. During the problem initialization phase, the planner is concerned only with tactical variables important to the mission.

When the problem is initialized, the planner begins by asking a series of "what if?" questions in an arbitrary sequence. ASTAP does not force the planner to follow a canned sequence of procedures. Instead, the planner may work through the solution process using the analysis and simulation tools provided in a sequence that seems natural to him. Aspects of the problem that require more analysis may easily be given the required attention emphasis.

When a promising plan (or partial plan) is obtained, the planner labels and stores it for future recall. An ASTAP plan consists basically of the following:

- Aircraft (A/C) tracks
- Ordnance release points
- Ordnance effects
- Survivability data

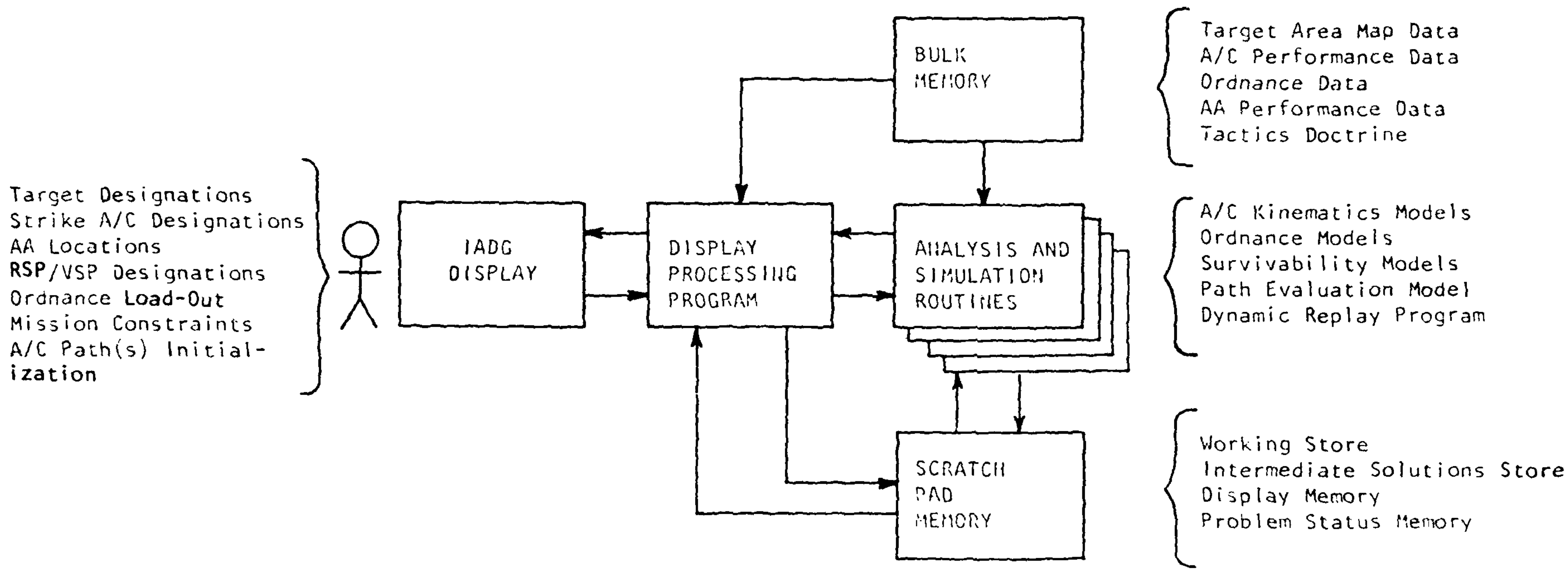


Figure 7. ASTAP Overview.

An example of the format is shown in Figure 8. Only a small number of display elements are presented to limit clutter. Various combinations of display elements (such as aircraft tracks, AA effectiveness contours, levels of map data) may be displayed by means of a special function key.

Dynamic replay of a plan shows the relative positions of the strike aircraft and their ordnance effects (blast debris, wind-driven smoke) in both time and space. The replay may be stopped at any point to examine relative aircraft positions, to assess safety aspects and examine potential problems due to arrival too early or late by designated aircraft.

Figure 8 is a representation of a strike involving three aircraft paths aimed at destroying buildings (T1) and a bridge (T2). The target area is defined by two AA installations whose radii of fire at the mission altitudes is shown. The double tracks indicate above-ground level (AGL) altitudes in excess of a designated threshold, e.g., 200 feet.

The basic plan calls for the aircraft along Path 1 to arrive first, staying outside the AA radii as shown. The predicted pop-up and ordnance release points (black squares) are computed by ASTAP using planner input/deployment maneuvers, aircraft speeds, ordnance type, and approach heading. The AA installations will have their attention diverted when aircraft along Path 2 do a pop-up along the ridge and first hit AA1. They continue along the shown path to neutralize the AA2 installation. The aircraft whose prime target is the bridge (T2) will follow closely behind. The intention is to time the strike so that the Path 3 aircraft are close enough to help swamp the air defenses, but not close enough to interfere or be exposed to interferences by the blast and smoke of Path 1 aircraft. The adequacy of the specific attributes of the plan are determined by the planner during dynamic display. Modifications to the plan are made to study the sensitivity of any part of the plan whose successful execution is dependent upon executing the plan within a timing or navation accuracy window.

This brief description gives just a limited glimpse of the possibilities of this concept. A partial list of ASTAP functions and capabilities is in Table 4.

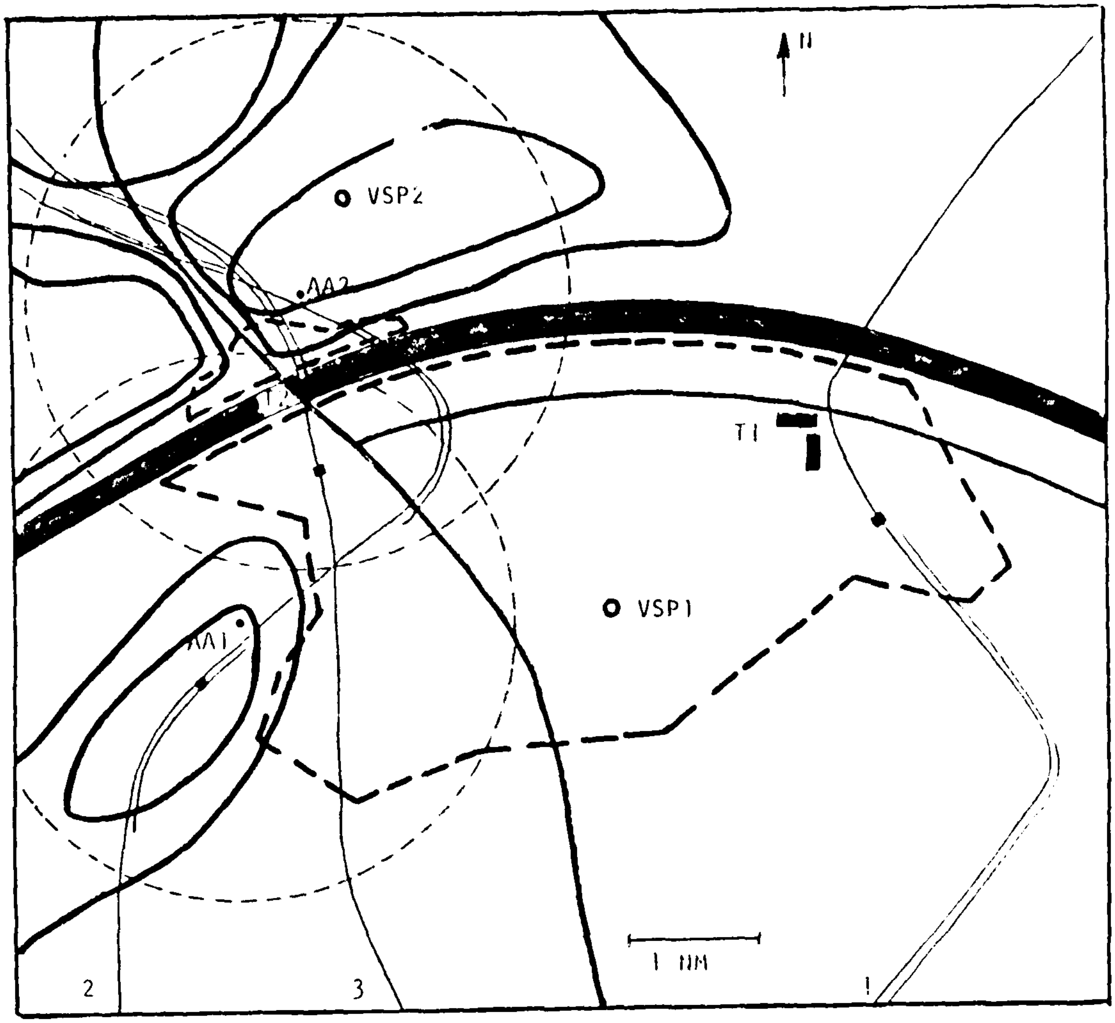


Figure 8. ASTAP Display Elements.

Terrain Contours

River

Road

Built-up Area

Low AGL A/C Path

High AGL A/C Path

Weapon Release Point

AA Limit

Visually Significant Point

NOTE: This shows examples of all the types of displayed information. They would normally not all be displayed simultaneously.

Table 4. ASTAP Functions.

- 3D path simulation from "distilled" large scale models and test results.
- Ordnance release point computation
- Ordnance effects computation
- 2D A/C path designation input and replay
- Survivability computation as a function of A/C paths and AA types and locations
- Computation of sighting data (ranges and bearings) to operator-designated points to coordinate target area navigation
- Computation and display of AGL values as a function of designated A/C speeds, horizon plane tracks, and type of ordnance deployment maneuvers.
- Computation and display of ordnance CEP's as a function of attack parameters
- Computation of fuel consumption as a function of changing drag count and A/C kinematics
- Dynamic replay of current and stored alternative plans with stop-action and add-on analysis capability.
- Automatic monitoring of user input constraints and constraint violation alerting

2.2.6 Hard-Copy Outputs

The planner will choose the preferred plan from those analyzed at the conclusion of a session with ASPS. The user may request a hard copy of the plan elements in the standard format on a knee-pad-sized sheet of paper. The printed output will include at least the following:

- Geographic coordinates and time of arrival at each turning point on the path
- Speed and altitude for each path leg
- Geographic coordinates, time of arrival, and altitude at a refueling point
- Loadout of weapons and external fuel tanks
- Coordinates of ordnance release points

3.0 INVESTIGATIONS OF CRITICAL ASPECTS OF ASPS DEVELOPMENT

Creation of make-believe terrain is acceptable for a laboratory version of ASPS, but a fieldable ASPS must use terrain data from real maps. Consequently, ISC conducted an investigation to learn how digital terrain data might be obtained and what algorithms are available for performing certain calculations on such data. The results of this work are given in Subsection 3.1.

Our ONR sponsor and air strike planners consulted by ISC expressed a strong preference for low-cost, highly portable ASPS hardware. ISC investigated desk-top, stand-alone computer and display systems and systems consisting of separate, but highly portable computer and display subsystems. The results of this work are in Subsection 3.2.

3.1 DIGITIZED MAPS

There are basically two methods of obtaining digitized terrain data:

1. The do-it-yourself method.
2. Use of digitized map products from the Defense Mapping Agency (DMA).

The first method is simply manually recording the terrain properties of a region on a cell-by-cell basis according to a predetermined data format for each cell. This is the only method available if DMA does not have a digitized map for an area of interest and the end user needs the map before DMA can meet a request for it. The Marine Corps training unit that operates the Tactical Warfare Simulation, Evaluation, and Anlaysis System (TWSEAS) at Camp Pendleton occasionally does its own digitizing of maps which are used in TWSEAS.

DMA digitized map products are derived from the Digital Landmass System (DLMS) Data Base (Reference 5). This data base is collected at two different levels which are defined as follows:

Level 1

1. Terrain: Relief information in DMA standard digital format with each cell three seconds of latitude (303.81 feet) by three seconds of longitude.

2. Cultural features: A generalized description and portrayal, in DMA standard digital format, of planimetric features. The Level 1 data base is intended to cover large expanses of the earth's surface and has relatively large minimum size requirements for portrayal of planimetric features.

Level 2

1. Terrain: Relief information in DMA standard digital format with each cell one second of latitude (101.3 feet) by one second of longitude.

2. Cultural features: A highly detailed description and portrayal, in DMA standard digital format, of planimetric features. The Level 2 data base is intended to cover small areas of interest and has small minimum size requirements for planimetric features.

An important consideration for ASPS development is the availability of already-developed computer programs for calculating topographic contours and for calculating masking effects of terrain around a given point. DMA has two applicable programs. One program, filed as "ULB 427" at the DMA Aerospace Center, St. Louis, accepts the DMA standard format for digitized terrain elevation data and produces a topographic contour map of the area. The other is a subroutine called "RTMASK" within the program filed as "ULB 437." RTMASK uses the output of ULB 427 and produces a plot that, given an observer's geographic position, will indicate the visibility afforded to that observer.

Information supplied by the DMA Aerospace Center led to several other sources of algorithms for calculating masking effects of terrain. One of these sources is work done by Pattern Analysis and Recognition Corporation, Rome, New York, for the Rome Air Development Center. Reference 6 documents their two FORTRAN algorithms for determining what areas on the ground are not detectable from an airborne radar due to terrain masking.[1] One of the algorithms uses a perspective projection approach and the other uses a ray tracing approach. DLMS Level 1 data from DMA are the input to their algorithms and the algorithms account for the effects of earth curvature and atmospheric refraction of the radar beam. Core requirements and execution speeds on an XDS Sigma V computer for several sizes of geographic areas are given in Reference 6.

Lincoln Laboratory at the Massachusetts Institute of Technology has a set of programs for calculating the effects of terrain masking on ground-based radars (Reference 7). These programs are written in PL/I for execution on an IBM 370 computer; they also use the DLMS Level 1 data as input. The terrain elevation data in the DLMS data base is converted from its matrix form to a polar form centered on the radar site. This is done for each elevation in the polar grid by interpolating from the four nearest elevations in the matrix grid. One drawback to the use of polar representations of data is the reduction in the density of data points as range increases from the origin. Compensation for this is made by increasing the azimuthal density of data points as range increases from the origin. The program then does a simple geometric calculation to determine those points in the polar grid for which the radar's line of sight is blocked. Inputs to the program include radar antenna height above ground and aircraft altitude above ground level. The program also accounts for earth curvature and atmospheric refraction of the radar beam. Program output indicates for each point on the polar grid whether or not the aircraft is masked.

Another terrain masking algorithm is available from a large-scale computer simulation of battles between ground based air defense systems and several weapons systems. This simulation is known as TACOS II (Reference 8).

---

[1] It is clear that, if a point on the ground is not observable from an aircraft, because of terrain feature obscuration, then the reverse is also true. Consequently, these algorithms are applicable to the problem of calculating terrain masking from ground defense sites. Radar refraction/ducting effects are not considered a significant advantage over visual reciprocity.

It was developed by the U.S. Army Combat Developments Command Air Defense Agency and Braddock, Dunn, and McDonald and it runs in FORTRAN on a CDC 6600 computer. The terrain masking algorithm is just one of many routines in TACOS II. Input to the algorithm is a matrix of terrain elevations taken at 500-meter intervals along each row and rows are spaced 500 meters apart. (This is not DLMS data.) The algorithm calculates the "Dominant Mask Function," i.e., the region of visibility within an air defense site's maximum range. On each radial from the site, the algorithm finds the elevation angle to each visible ridge and determines the ground distance beyond each ridge and along the radial where masking occurs. Figure 9 illustrates the idea. The algorithm accounts for curvature of the earth.

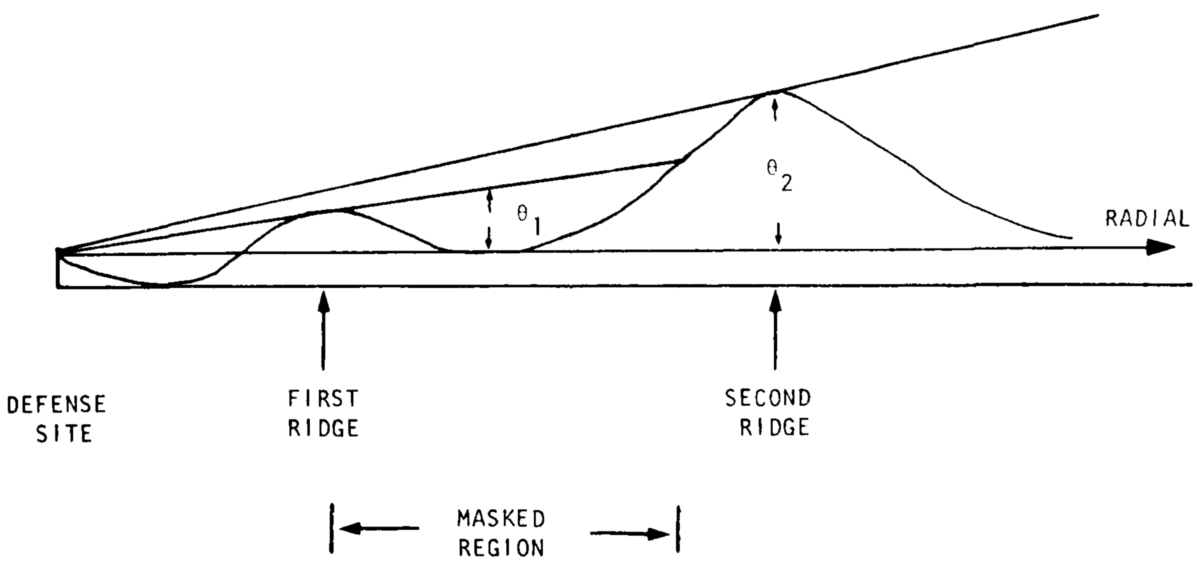


Figure 9. Illustration of a Masked Region Along a Radial from a Defense Site.

Thus, the results of our investigation of digitized maps show that:

1. There is a large and growing body of DLMS Level 1 digitized terrain data available from DMA which is suitable for use in ASPS.

2. There are several existing algorithms for calculating topographic contours and for calculating masking effects of terrain around a given point. It is reasonable to believe that algorithms suitable for ASPS can be devised by modifying ones already in existence.

3.2 INVESTIGATION OF HARDWARE ALTERNATIVES FOR ASPS

ISC investigated the products of 16 manufacturers of graphics terminals to find the best hardware available for the desktop version of ASPS. We restricted our investigation to the relatively large firms to better insure future availability of both service and parts. The major considerations in our analysis were:

- computation speed
- cost
- coding capabilities
- peripheral availabilities
- expandability of system
- ease of displaying and programming dynamics
- ease of entering map data

3.2.1 Computation Speed

The algorithms that are needed for the optimization modes of ASPS ("Rising Water" and other dynamic programming algorithms) involve a great deal of computation. Execution speed, therefore, is of great importance. Most systems which ran only the BASIC computer language were eliminated. This is because BASIC is an interpretive language, i.e., every line of code must be re-interpreted each time it is executed and this increases computation time considerably. FORTRAN, on the other hand, is a compiler language which is translated into machine language and, thus, each line of code is interpreted only once.

ISC previously developed defense penetration planning aids which used both "Rising Water" and other dynamic programming algorithms. These programs were implemented on a Varian V-73 computer. The execution speed of the V-73 was used as a benchmark to determine the relative speed of the various systems examined. We wrote a program which consisted of code similar to that used in the aforementioned algorithms. These programs were run on the V-73 and on the other systems examined. The V-73 executed the program in the fastest time (20 seconds). The other times are listed in the summary sheets of each system's characteristics which are shown below.

3.2.2 Raster vs. Vector

Two basic display technologies are available for the desktop versions of ASPS, namely, vector and raster graphics. Vector graphic images are produced by a "gun" which draws on a tube at very high speed. Raster graphics produces an image consisting of many points or pixels (currently as many as a million), each of which may be chosen to be a certain color. Using vector graphics, it is simple to create symbols and change their location frequently enough to provide smooth movement on the display. It is also an easy matter to blink these symbols or change their color. Animation is possible in raster graphics, but some fluidity of motion may be lost. Raster graphics facilitate colored background and shading where only lines and circles may be drawn in vector graphics. We principally examined raster graphic systems since this technology is more widely used in both the military and non-military sectors than is the vector-based technology. Also, raster systems currently sell for $30,000 - $160,000, while the vector-based systems generally begin at $60,000. Color and shading provide an important coding dimension, and all raster systems supply this.

3.2.3 Peripherals

ASPS requires an input device which allows the user to designate a location on the screen with ease. A joystick or trackball is a good choice since either facilitates a continuous stream of updated locations on the screen to aid in the updating of a path. Special function keys would also be necessary to allow the user to change modes in ASPS. These keys would be clearly labeled, and would have lights associated with them which can be turned on when activated.

3.2.4 Map Data

We have explored various methods for displaying map data. One method requires digitizing map features and placing it in a data file. This file is subsequently read and displayed on the screen. This method can produce good terrain detail, but is slow to display and requires an enormous amount of storage. A second method employs a 35mm slide of the area which can be displayed through special equipment. This may be impractical, if not impossible, when the computer does not have the capability to accept the extra equipment required for this method. A third method is to store topographical contours in coded form in a data file. This seems to be the most practical and manageable method for displaying the region in question and is certainly adequate for ASPS. Using this last method, the terrain data can readily be held in a data file and displayed equally well on all systems under review.

3.2.5 Dynamics

The current ASPS work does not require a great deal of dynamics on the screen. The users need to alter a path which does not require an extensive amount of dynamic capability. However, the ability to easily program dynamics may well have major importance for any follow-on work for ASPS.

3.2.6 Comparisons Among Five Candidate Systems

With these considerations in mind, ISC compared five candidate systems:

- Tektronix 4054
- Hewlett-Packard 9845C
- Genisco GOT-3000
- Megatek 6250
- Megatek 7250

(Genisco GOT-3000, Megatek 6250, Megatek 7250) coupled with the Digital Equipment Corporation (DEC) MINC 11/23 as a host computer

3.2.6.1 Tektronix 4054. This system is a stand-alone, interactive graphics computer. The single, compact console consists of a 19-inch, monochromatic storage tube, a full ASCII keyboard, twenty special-function keys, two thumbwheels (dials which are used to location positions on the screen), and a tape drive.

EXECUTION SPEED: The slowest of the four[1] central processing units whose speeds were measured (more than twice as slow as the MINC 11/23). The speed, however, would be sufficient for ASPS. Benchmark program execution time: 121 seconds.

COST: The entire system with an additional two floppy discs costs approximately $30,000.

CODING: The greatest weakness of the system. We found lack of color to be a great disadvantage. Textured lines (e.g., dot-dash, dot-dot-dash) do not have the capability of color-coding to convey complex "scenes" of information.

PERIPHERALS: The essential peripherals are built into the basic console. Floppy disc drives, graphics tablets, and printer can be easily added.

EXPANDABILITY: Since the Tektronix was designed as a stand-alone system, it will not accommodate more than one work station.

DYNAMICS: This vector graphics system provides fluid movement of objects. However, it is difficult to have simultaneous movement of a number of objects.

MISCELLANEOUS: Editing and programming features are cumbersome and would add to the programming cost of ASPS if this system were used.

---

[1]The benchmark routine was timed on the Varian V-73, MINC 11/23, Tektronix 4054, and H.P. 9845-C.

3.2.6.2 Hewlett-Packard 9845-C. The Hewlett-Packard (H.P.) 9845-C is a compact, stand-alone desktop computer with a color graphics raster display. The standard system comes with built-in hard-copy device, two tape drives, a keyboard with 8 special function keys (with use of the shift keys, this is effectively raised to 32), and a lightpen. Additionally, there are 8 software programmable keys which are conveniently located just below the CRT to allow labeling on the screen. Although this computer may only be programmed in BASIC, its computation speed is adequate for the requirements of ASPS.

EXECUTION SPEED: Half as fast as the MINC 11/23, but adequate for ASPS. Benchmark program execution time: 86 seconds.

COST: This system with two floppy discs costs approximately $46,000.

CODING: There are virtually an unlimited number of colors available of which 16 may be used at any one time. The resolution of 560 X 455 provides more than sufficient clarity for ASPS. Shading and filling of polygons is easily accomplished in the software.

PERIPHERALS: Floppy disc is available for an additional $5,000. A trackball is not available from Hewlett-Packard, but may be obtained from a second source for $3,000-$5,000. A graphics tablet is available from H.P. which can be used in place of a trackball or joystick for about $3,000.

EXPANDABILITY: This desktop unit was not designed as an expandable system. Only one work station could be supported from each computer.

DYNAMICS: The lack of easily programmable dynamics is a disadvantage of this system. None of the demonstration programs by H.P. showed any dynamics. This capability was not of high priority in H.P.'s design and, hence, programming dynamics is rather difficult and, in some complicated cases, impossible.

MISCELLANEOUS: The lightpen does not provide a facile means of altering a path. The editing capabilities on the H.P. 9845 are excellent and will facilitate programming.

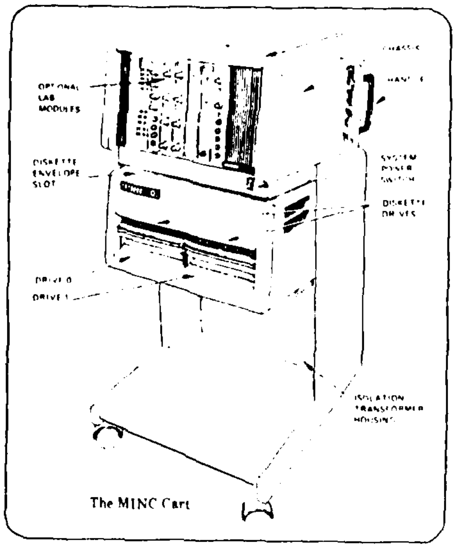
3.2.6.3 Systems Using the DEC MINC 11/23. The next three graphic systems use the DEC MINC 11/23 as a host computer (see Figure 10). The MINC 11/23 can support two display monitors working simultaneously. FORTRAN is programmable on this machine and good editing features are available. It gave the fastest result of the benchmark routine of all systems under consideration, about 45 seconds.[1] A fast floating-point processor will soon be available which will further increase its speed and efficiency. The MINC 11/23 is extremely lightweight and portable, coming standard on a mobile cart where the components reside. Adding components is very simple and designed in such a way so that the user can't make a mistake. Data files could be created on other DEC equipment, such as the VAX, and used on the on-board system, which increases its flexibility. Also, the software developed for the MINC is compatible with the VAX.

There are some disadvantages to a component system, such as the MINC 11/23 combined with display monitor. Having a separate host computer and display monitor would require separate maintenance contracts and cabling between the components. Cabling would add a slight cost and get in the way. The operating system for the MINC is not resident in core, but is read in via a floppy disk. Much precaution must be observed to protect these diskettes. The MINC only accepts a Q-Bus connector to its peripherals. Most of the display terminals reviewed require a Uni-bus connector, but converters are readily available. The problems of the MINC 11/23 are really quite minor and would not hamper the operation of the ASPS project.

Megatek 7250 and DEC MINC 11/23. This combination is well suited for current and add-on ASPS work. The ease of programming dynamics is a great advantage to this system. The 512 x 512 resolution is quite adequate

---

[1]Varian V-73 is not under consideration because of its size, but was used as a comparison for the benchmark times.



The MINC Cart

Figure 10. DEC MINC 11/23.

for the level of detail required for this project. This system has a pan and zoom capability built in the hardware. The analysis below is for the 7250 with a keyboard, joystick, 16 lighted function switches, and 8 control dials.

COST:	MINC 11/23	Approximately $18,000
	MEGATEK 7250	Approximately $38,000
		Approximately $56,000

CODING: Up to 16 colors may be used at one time. Filling polygons is cumbersome and must be done in software. A hardware fill capability will soon be available.

PERIPHERALS: Hardcopy device (black and white), data tablet, and pick module (similar to a lightpen) are among the available options.

EXPANDABILITY: The heart of this system, the MINC 11/23, is well suited for expandability. Two work stations could be run from the computer.

DYNAMICS: The capability to easily program complex dynamics is the forte of the Megatek 7250. The data link from the MINC to this display is fast enough to support independent fluid movement of multiple symbols on a complex tactical decision aid.

Megatek 6250 and DEC MINC 11/23. This is a cheaper and less powerful version of the Megatek 7250 combination. The most noticeable loss is the complexity of dynamics which can be displayed. The MINC talks to Megatek 6250 over an RS-232 interface instead of the unibus. The result of this is that the complexity (number of vectors) which can be updated every frame is cut back drastically. However, the dynamic capabilities of the Megatek 6250 are adequate for the current version of the ASPS project although complex scenarios with multiple objects moving simultaneously might cause some difficulties.

The 6250 comes standard with a keyboard and joystick. The special function box is not an available option for this model. Another drawback of this sytem is that the only available display screen is just 13 inches across the diagonal.

COST:	MINC 11/23	Approximately $18,000
	MEGATEK 625	Approximately $20,000
		Approximately $38,000

Genisco 3000. This display system is a competitor of the two Megatek systems that have been previously reviewed. It would require a Q-Bus-to-Unibus converter to run off the MINC 11/23 computer. This is readily available. There are several monitors available with varying refresh rates and resolutions. The 512 X 512 resolution monitor has a maximum refresh rate of 30 Hz, whereas, the 640 X 480 resolution monitor can be increased to 40 Hz. There are other monitors available with better resolution and faster refresh rates, but the aforementioned monitors would be adequate for ASPS. If very complex movement is involved, then flickering or rough movement of objects might occur. Also, while moving an object around the screen, a ghosting effect (i.e., a slight trail of the image being moved) can occur. This is not envisaged to be a problem in ASPS.

The Genisco system can be programmed by either referencing pixels (used on raster systems) or specifying vector lines (used on vector systems). Genisco is best suited for image-processing, although dynamics can be easily performed, but not with the same clarity as a vector system. For the ASPS project, the dynamics have a major importance, while the image-producing capability has a smaller role and, hence, the Genisco 3000 would have more potential than what is required.

The Genisco 3000 has virtually unlimited color shading available on a single board which contains four bit planes. A trackball, joystick, and alphanumerics keyboard with sixteen lightable special function keys are easily added. These special function keys can be used in conjunction with the shift and control keys to have 64 distinct functions. This system can be expanded to two monitors, if required.

COST:	MINC 11/23	Approximately $18,000
	GENISCO 3000	Approximately $25,000
		Approximately $43,000

3.2.7 Recommendation

The Megatek displays are more suited to this project than is the Genisco because of the way dynamics are accommodated. We have not seen Genisco demonstrations of the type of dynamics which are required for ASPS and would be reluctant to recommend this display system.

Of the two Megatek systems, we recommend the 6250 if the current ASPS concept is to be implemented as now written. The 6250 could well support the requirements of the current ASPS system, but might not be suitable for follow-on work which involves more complex dynamics. The 7250, while more expensive, permits all the dynamics needed for a tactical decision aid. The special function key box which is available with this model is the input device most suited for this application. Of all the systems investigated, the Megatek 7250 most closely fits the needs of the current ASPS work, and would not restrict adding capabilities in the future.

All five of the systems examined are state-of-the-art systems. All except the Tektronix 4054 could be used for the on-board ASPS system. The Hewlett-Packard is a very impressive system. Its compactness and easy transportability is a true benefit as compared to multiple component systems linked to the MINC 11/23. However, difficulty of displaying dynamics, and the lack of a trackball or joystick make the H.P. a less desirable system than those driven by the DEC computer. The expandability of the DEC computer system may be important for ASPS follow-on work.

4.0 RECOMMENDATION

Procedures for planning Naval air strike missions are mainly done by manual processes at the present time. Planners spend much of the available planning time doing tasks such as:

1. Calculating fuel consumption for alternative routes and weapon loadings.
2. Measuring ranges and bearings from turning points to radar significant points or visually significant points.
3. Measuring courses and latitude and longitude at points on a map.
4. Recording a plan on a standard form.

Planners do not currently have the time to calculate terrain masking regions for different AGL's around defense sites nor are they able to calculate the effect of terrain masking on the probability of survival. All of the above tasks can be done by computer software whose use is controlled by the planner.

Development of a highly interactive ASPS is recommended because ASPS will enable planners to use their time more efficiently than is possible with current manual procedures and because cost of ASPS development and purchase will be low. The ASPS concept described in Subsection 2.2 will allow the planner to do all that is currently done in much less time per planning session. Alternatively, with ASPS the planner can evaluate many more alternative plans in the same amount of time that is currently used per session. Further, certain ASPS features will enable the planner to do analyses that are not practical at present. An example is the quantitative evaluation of terrain masking on alternative plans. Another reason why ASPS will be attractive to strike planners is that it is essentially a powerful calculation and display assistant. The planner's judgment and reasoning is in control throughout the planning process; ASPS is not configured to force any take-it-or-leave-it decisions on the planner.

Cost of ASPS development will be low for several reasons. The state of computer technology is advancing so rapidly that exceptionally powerful, low cost desktop computers are just coming on the market. (An example is a desktop VAX computer made by Digital Equipment Corporation; this model will be available early in 1982.) The Land Minefield Planning Aid System (LAMPAS) is a Marine Corps program which is taking advantage of this advance in technology by incorporating sophisticated models in a desktop computer and display system. Consequently, our judgment is that hardware cost for a production version of ASPS would be only about $60,000.[1] There are two other reasons why cost of ASPS development would be low. One is that much of the design work has already been done. The other is that much of the software, including algorithms for determining terrain masking around defense sites, has already been developed.

---

[1] This is a non-GSA price for a single unit.

REFERENCES

1. Irving, G.W., et al. Experimental Investigation of Sketch Model Accuracy and Usefulness in a Simulated Tactical Decision Aiding Task. Report No. 215-3, Integrated Sciences Corporation, Santa Monica, California, May 1977.

2. Walsh, D.H. and Schechterman, M.D. Experimental Investigation of the Usefulness of Operator Aided Optimization in a Simulated Tactical Decision Aiding Task. Report No. 215-4, Integrated Sciences Corporation, Santa Monica, California, January 1978.

3. Schechterman, M.D. and Walsh, D.H. Comparison of Operator Aided Optimization With Iterative Manual Optimization in a Simulated Tactical Decision Aiding Task. Report No. 215-6, Integrated Sciences Corporation, Santa Monica, California, July 1980.

4. Rebane, G.J. and Walsh, D.H. Concept for an Air Strike Planning System. Technical Memorandum 330-1, Integrated Sciences Corporation, Santa Monica, California, February 1981.

5. Defense Mapping Agency. Produce Specifications for Digital Landmass System (DLMS) Data Base. PS/ICD/100; PS/ICE/100; PS/ICF/100; PS/ICG/100, Aerospace Center, St. Louis, July 1977.

6. Jackson, R. and Reed, C. Terrain Shadow Prediction. MRS^3 Working Paper No. 29, Pattern Analysis and Recognition Corp., Rome, New York, 22 February 1977.

7. Delaney, J.R., et al. Description of Terrain Masking Prediction Software. Report CMT-4, Lincoln Laboratory, MIT, 20 March 1980.

8. Braddock, Dunn, and McDonald. TACOS II Documentation, Volume IIIA: FRAG1A, FRAG1B, and FRAG1C User/Planner Manual. BDM Report A-13-74-TR-R1, Albuquerque, New Mexico, 15 May 1974.

APPENDIX A:

SOFTWARE STRUCTURE

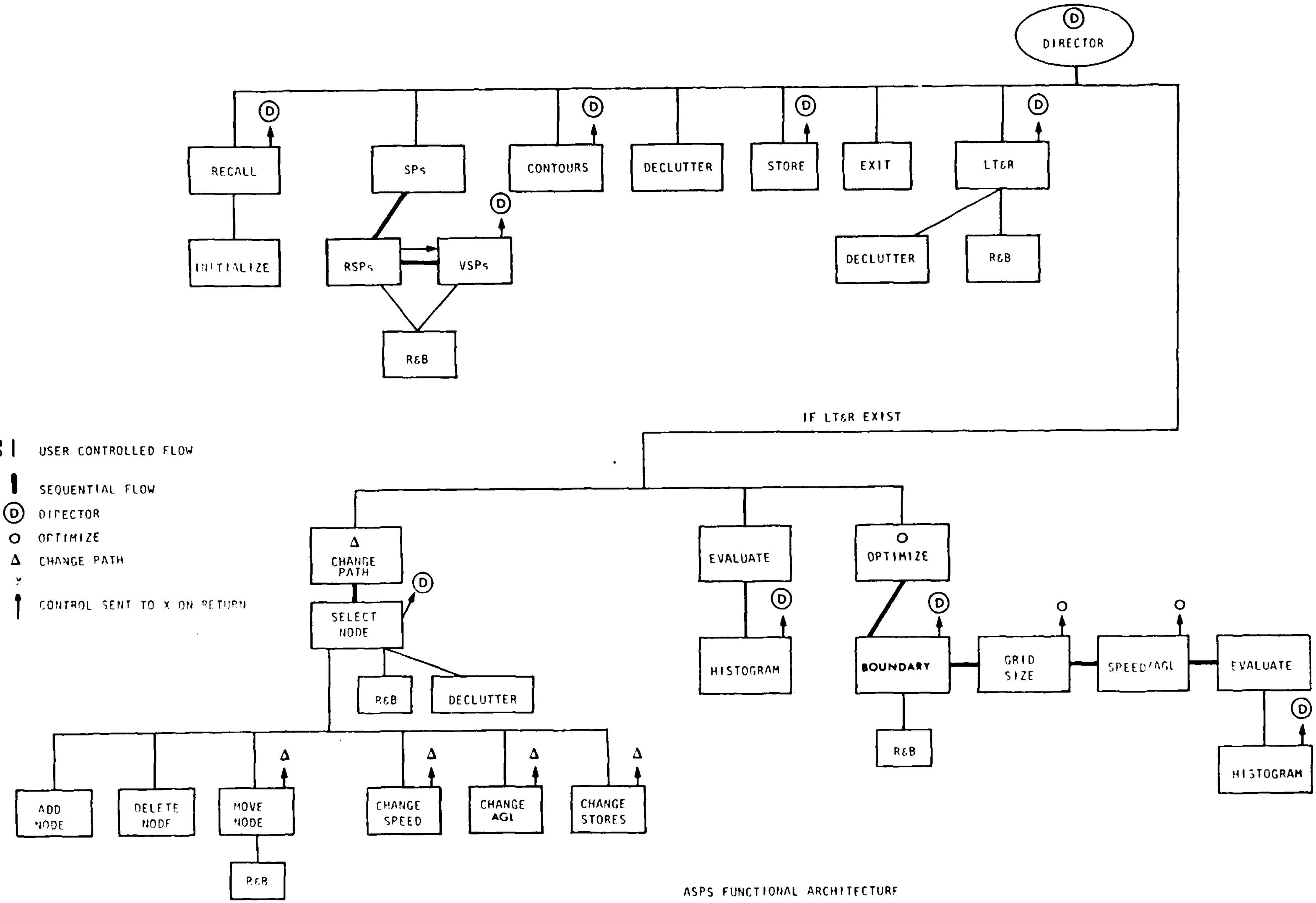
APPENDIX A: SOFTWARE STRUCTURE

A block diagram of ASPS's modular software structure is given in Figure A-1. For each box listed directly below the "Director" routine, there is a special function key as shown in Figure A-2. Routines indicated by boxes below the first tier boxes (special function keys) are manipulated through displayed menus and use of the ACCEPT and REJECT keys, number pad and/or trackball. The software implementing each function is self-contained to facilitate modifications, integration of new functions, and understanding the flow. The Director guides the top level flow of the program to the proper subroutines and ignores improper or untimely commands. Only those special function keys which are lighted at a given time are active. An example of an untimely command is attempting to change a path before a path exists. If no path exists, the "Δ CHANGE PATH" key is not activated (lighted).

The program consists of three overlays. When going between overlays, the flow always goes through the Director routine. The Director guides the flow to the proper overlay when the user presses an activated special function key. The three overlays contain modes which correspond to the following special function keys:

Overlay 1	Overlay 2	Overlay 3
Director	Change Path	Contours
Exit	Significant Points	Evaluate
Recall		Launch/Target/Recovery
Store		Optimize

Each mode has a specific task to perform and is entered via the Director. The user should know the purpose of each mode and how it is used. Instructions and a title for each mode are shown on the display to aid the operator. The subroutines "Evaluate" and "Optimize" do not currently perform specific mathematical computations. From each mode, the user can easily return to the Director Mode.



ASPS FUNCTIONAL ARCHITECTURE

Figure A-1.

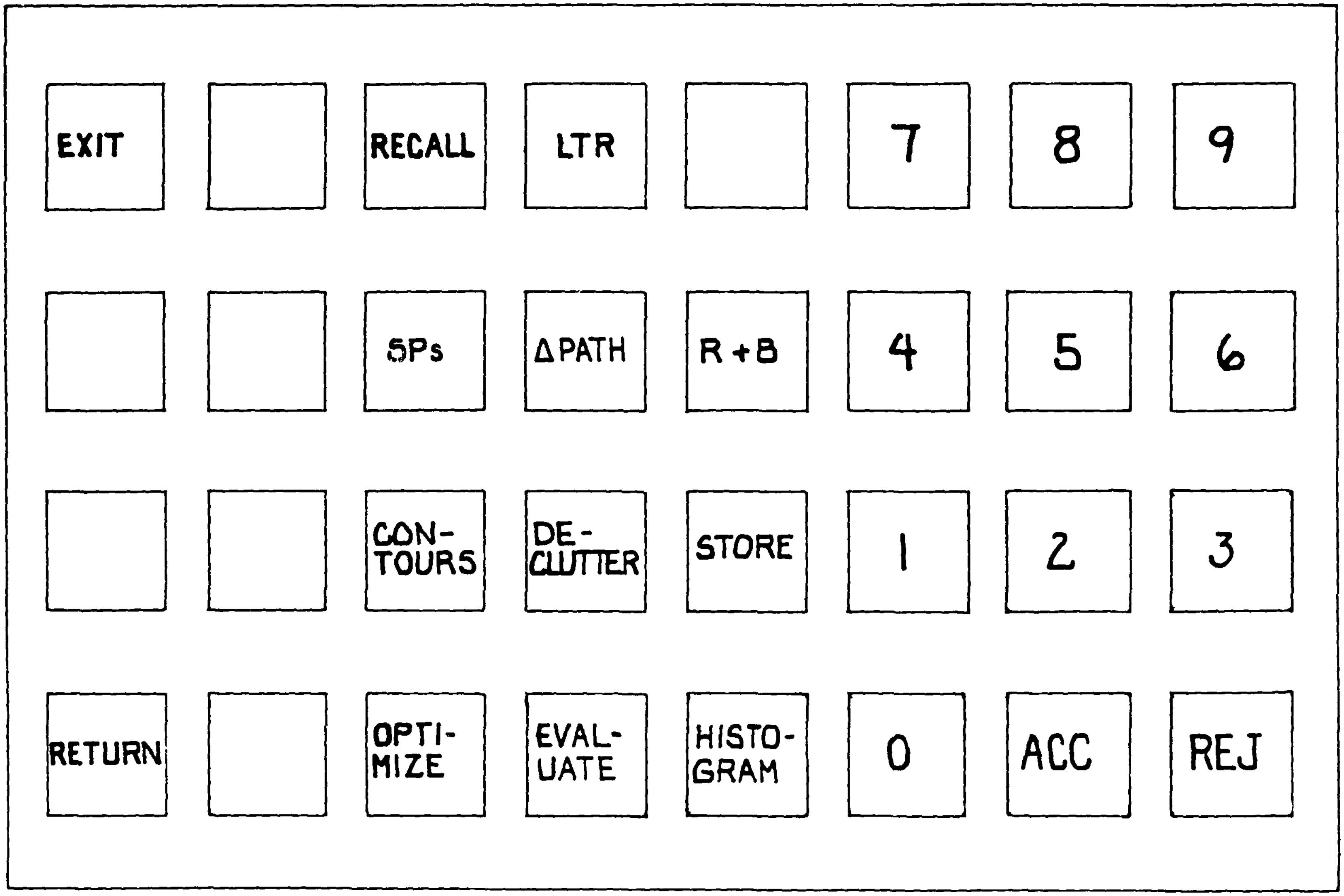


Figure A-2. Function Key Setup.

In addition to the modal keys, there are two special function keys used as aids for the operator, namely DECLUTTER and RANGE/BEARING. The DECLUTTER key allows the operator to call up or delete the display of significant points, defense capability contours, or both. This key is active while the program is in the Director, Change Path, Significant Points, and Launch/Target/Recovery Modes. The RANGE/BEARING key allows the user to get a range and bearing between two points. It is active while in the Change Path, Significant Points, and Launch/Target/Recovery Modes.

There is a bookmark routine which identifies and allows for the erasing of contour entities from the display file. Each time the Contours Mode is called, the previous entity numbers used for the contours are eliminated from the display file and then reused. The bookmark routine is not used when the DECLUTTER key is pressed to make contours disappear and reappear. The bookmark routine is only used when entering the Contours Mode. This allows a variable number of contours to be drawn and allows a variable length for each contour. The display file will not contain extraneous contours and, hence, the program runs more efficiently. All other entities have fixed length and are identified in the subroutine "Setup."

APPENDIX B:

DATA FILE STRUCTURE

APPENDIX B: DATA FILE STRUCTURE

The bookkeeping of the work done on ASPS is stored on disk in the data file "Admins." This is a data file which keeps track of the following information:

1. Number of maps and scenarios available
2. Last map and scenario recalled
3. Current scenario being worked on
4. Figure of Merit value for each scenario
5. Optimal scenario for each map thus far.

A description of the contents of the Admins file is given in Figure A-3.

For each scenario, there is a disk file with pertinent information to be read into the Master Data File (MDF). The MDF is set up such that the first fourteen words are pointers to their respective sections. This was done to allow maximum flexibility in the MDF array. Figure A-4 shows the MDF header with its set of pointers to specific locations within the MDF. Figures A-5 through A-13 describe each section of the MDF.

Some of the entries in the MDF are fixed, while others are supplied during the execution of the program by the operator. The permanent features include terrain, sensor positions, SAM site positions, AA site positions, origin and scale, elevations and character strings, and all defense capability contours. The information supplied by the user includes VSP/RSP positions, stores being carried, and the flight path.

The contents of the Admin file prevent user access to a Master Data file until a scenario has been stored in it. When a scenario is STORED by the operator, the Admin data is updated, but not stored onto disk until the operator presses EXIT.

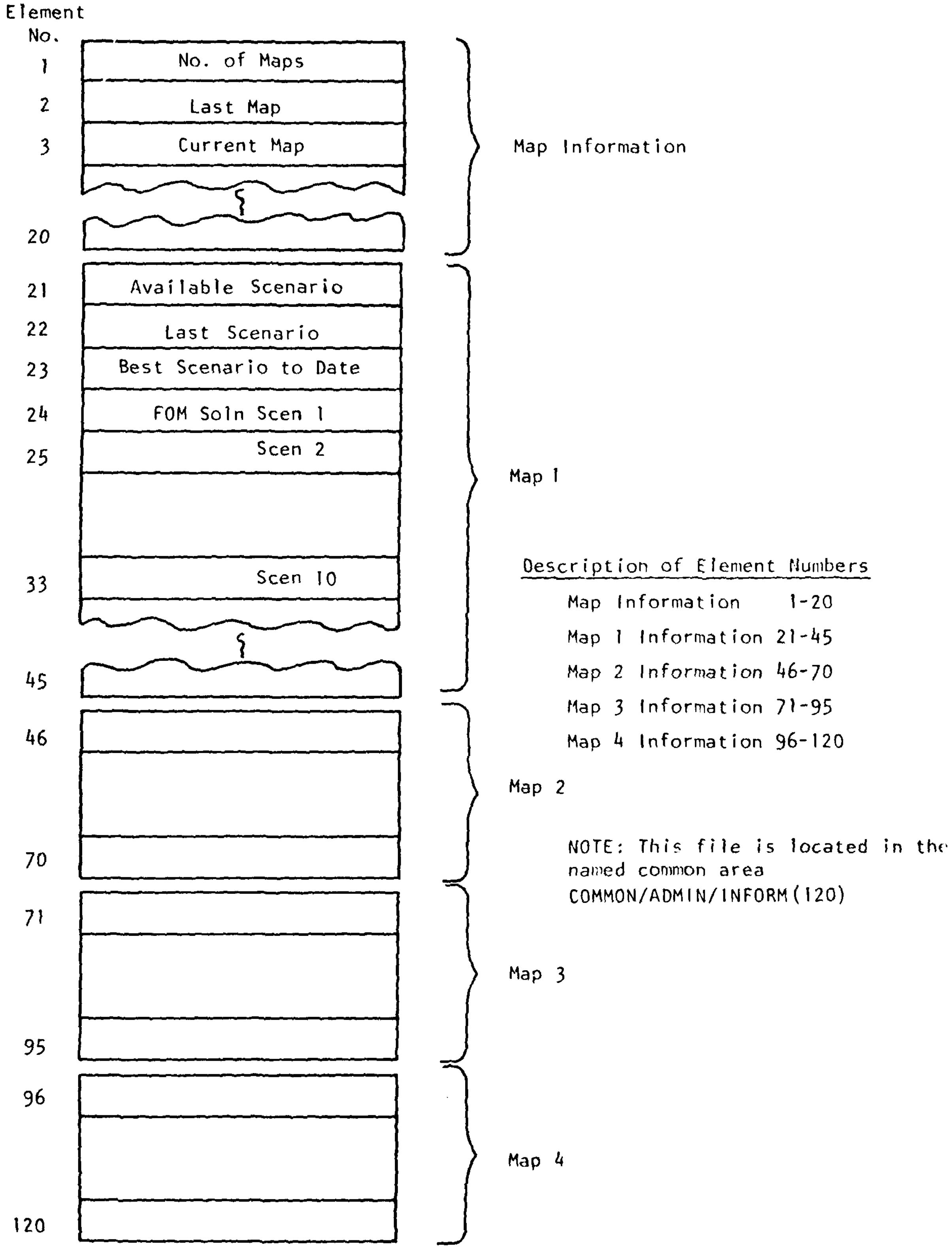


Figure A-3. Contents of Admins Data File.

1.	Terrain pointer
2.	Sensor pointer
3.	Surface to Air Missile (SAM) Pointer
4.	Character string and elevation pointer
5.	Lat., Long. and Scale pointer
6.	RSP pointer
7.	VSP pointer
8.	Path pointer
9.	Stores pointer
10.	Sensor Contour Pointer
11.	SAM Contour
12.	Anti-Aircraft (AA) Contour
13.	(AA) Pointer
14.	Scratch pointer

Figure A-4. Data Format of the Master Data File Header.

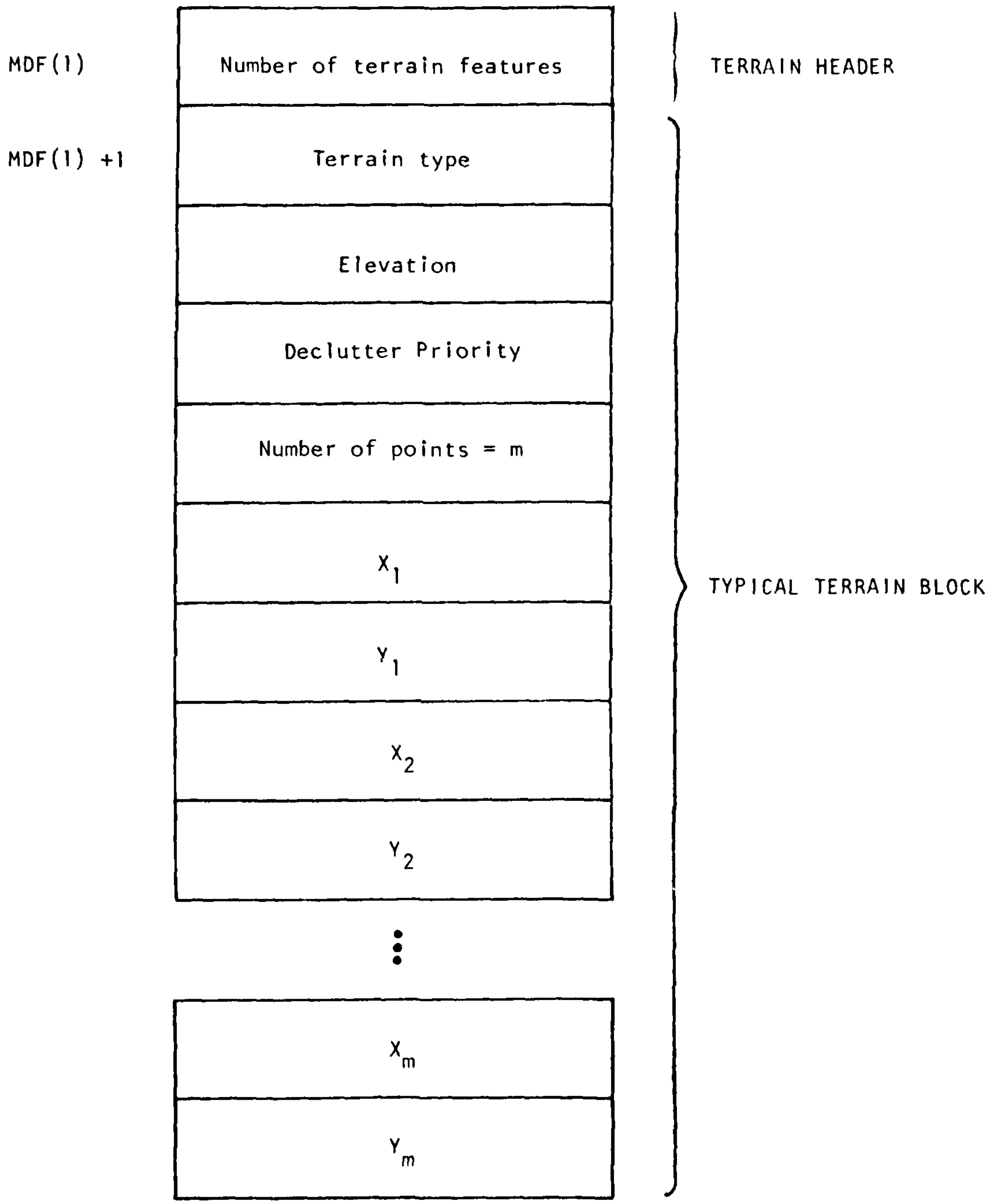


Figure A-5. Terrain Data Format.

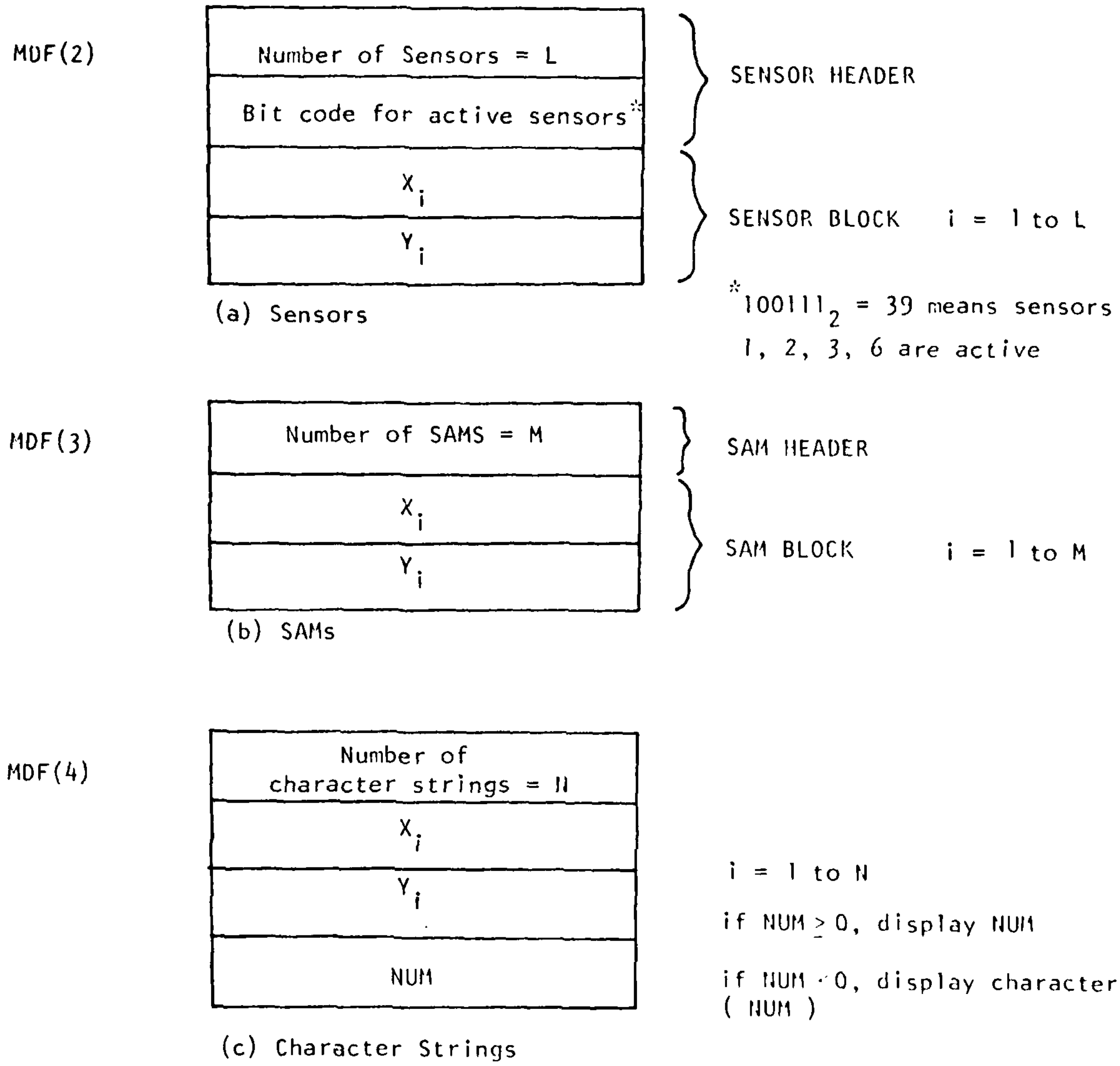


Figure A-6. Sensor, SAM, and Character String Data Formats.

MDF(5)	
	lat. in minutes
	long. in minutes
	width of screen in nautical miles

(a) Latitude/Longitude Format

MDF(6)		
	Number of RSPs = N	RSP header
	X_i	$i = 1$ to N
	Y_i	

(b) RSP Format

MDF(7)		
	number of VSPs = M	VSP header
	X_i	$i = 1$ to M
	Y_i	

(c) VSP Format

Figure A-7. Data Format for Latitude/Longitude Position and Significant Points.

MDF (8)	1	No. of Legs = N	Header
	2	No. of Target Nodes	
	3	(not used for now)	
(Launch) 1st Node and Leg	4	X - 1st Node	
	5	Y - 1st Node	
	6	AGL - 1st Leg	
	7	Speed - 1st Leg	
	8	Drag - 1st Leg	
2nd Node and Leg	9	X - 2nd Node	
	10	Y - 2nd Node	
	11	AGL - 2nd Node	
	12	Speed - 2nd Leg	
	13	Drag - 2nd Leg	
3rd Node and Leg	14	X - 3rd Node	
	15	Y - 3rd Node	
	16	Speed - 3rd Node	
Nth Node and Leg			
Recovery Node		X - Coordinate	Final 2 words for Recovery Position
	5(N+1)	Y - Coordinate	

Maximum of 10 blocks each with 5 words per block

IPATH (Entity 300)

DIM IPATH = 55

NOTE: This information is placed in the IPATH array found in the common area: COMMON/ROUTE/IPATH (65)

Figure A-8. Data Format of Path Information.

MDF (9)

No. of 500 lb. Bombs
1000 lb. Bombs
Missiles
Fuel Tanks

Figure A-9. Stores Data Format.

Each word is a pointer for each AGL and for each sensor and composite.

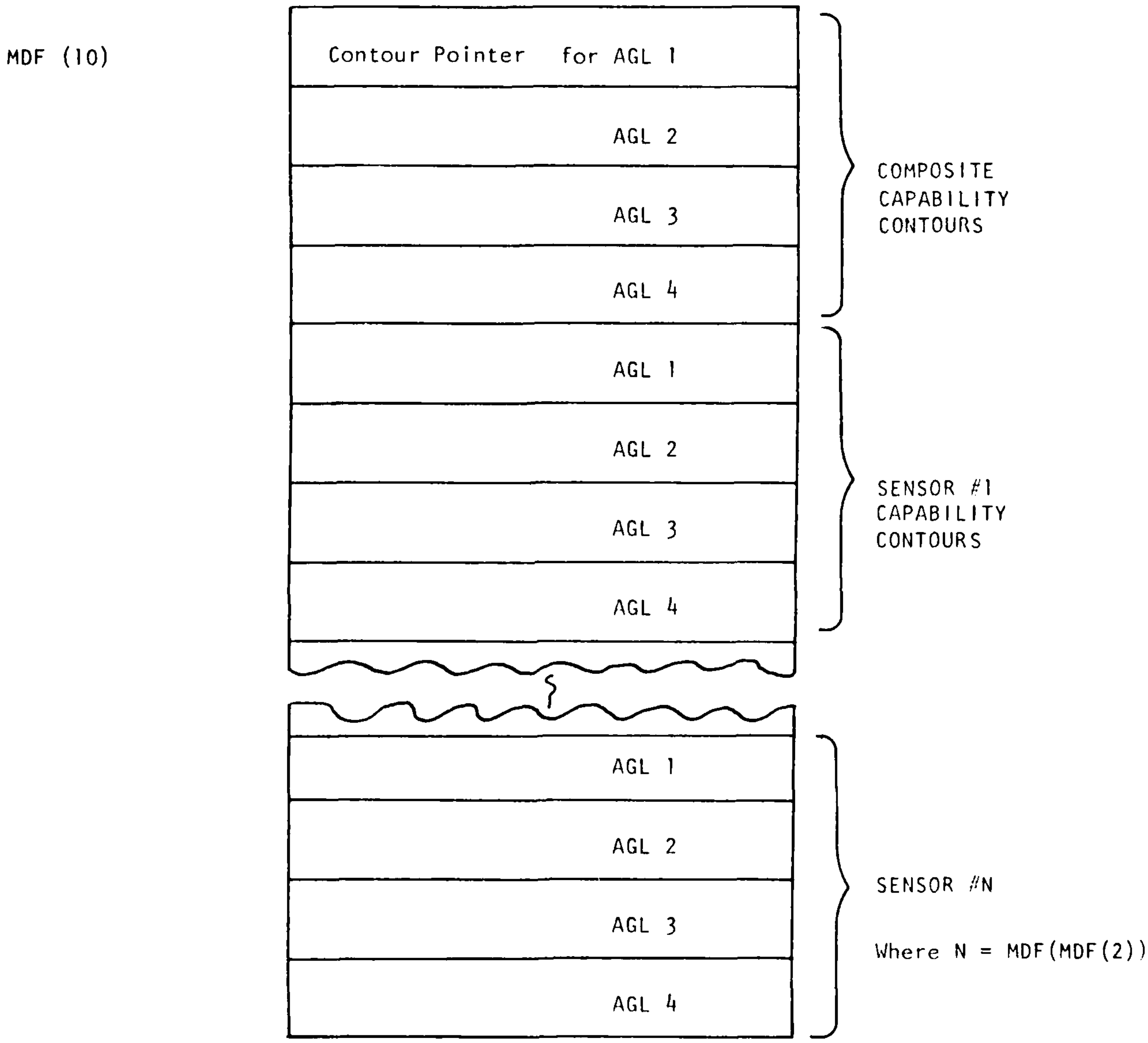


Figure A-10. Data Format of Sensor Contour Pointers.

For each AGL and each Sensor (including the Composite) the format is as follows:

No. of disjoint figures = M

No. of points in figure 1 = N_1

X_1

Y_1

⋮

X_{N_1}

Y_{N_1}

No. of points in figure 2 = N_2

X_1

Y_1

⋮

X_{N_2}

Y_{N_2}

No. of points in figure M = N_m

X_1

Y_1

⋮

X_{N_m}

Y_{N_m}

Figure A-11. Data Format of Sensor Contour Points.

MDF (11)

SAM capability is modeled as a wide-angle cone with apex at the site.

"Radius"means horizon-plane radius of SAM capability at AGL elevation.

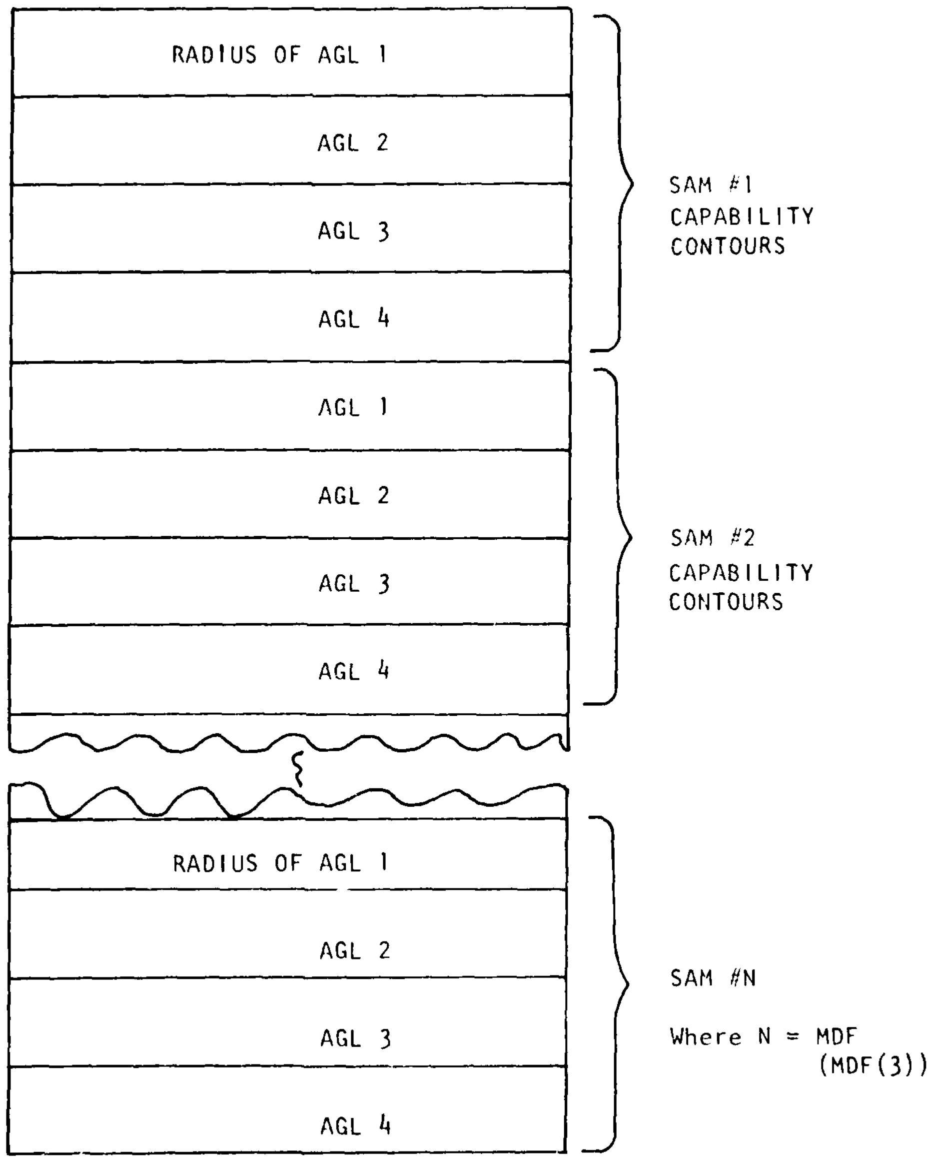


Figure A-12. Data Format for SAM Contours.

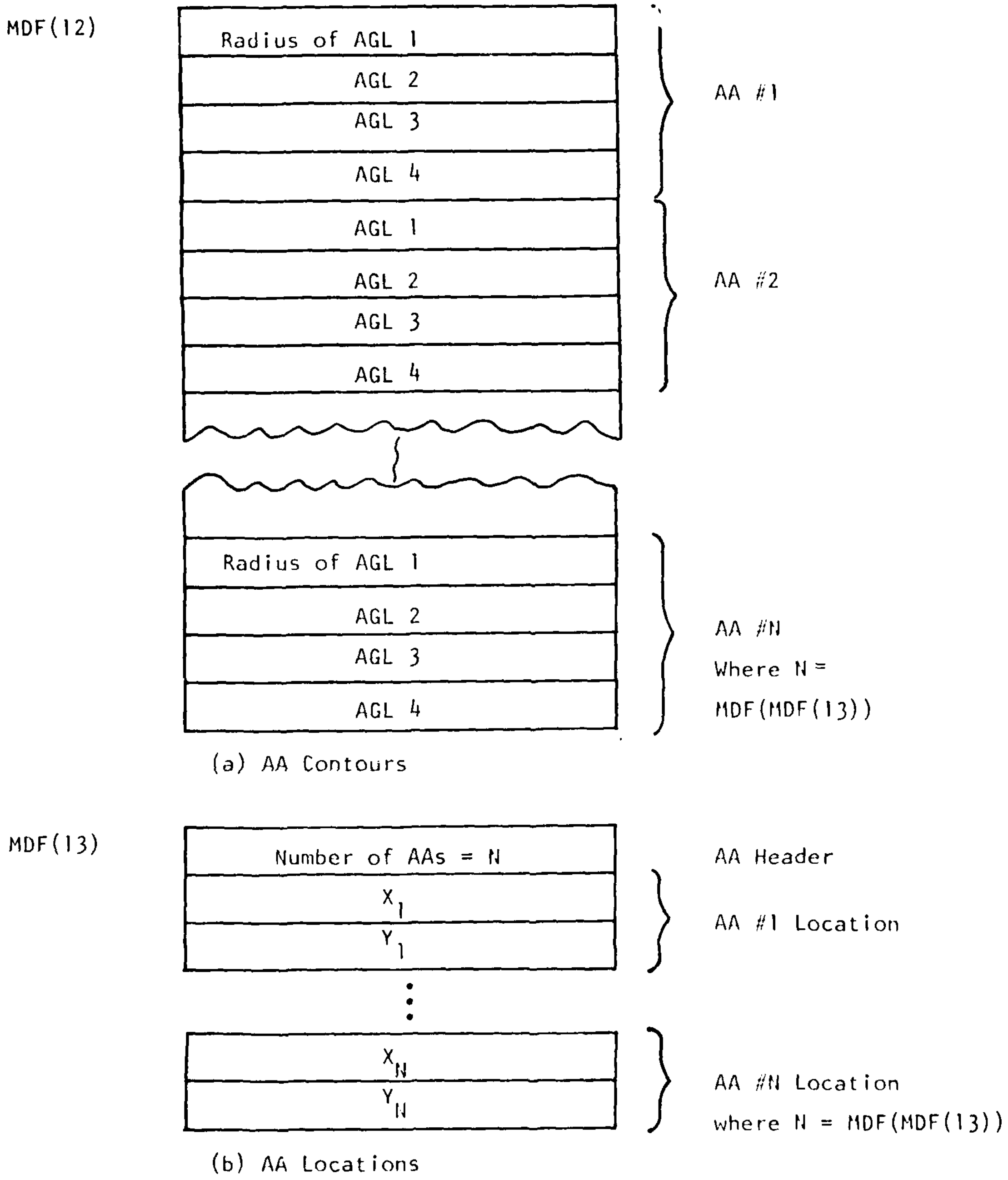


Figure A-13. Data Format for AA Contours and Locations.

APPENDIX C:

SOFTWARE LISTINGS

```
          COMMON /ADMIN/ INFORM(120)
          COMMON /MISTER/ MDF(2400)
          COMMON /ATT/ IATT(12),I[illegible]R
          [illegible]IT(9),[illegible](3),IFLOW(5)
          [illegible],IMPR(18)
          COMMON /[illegible]/ [illegible]
          LOGICAL L[illegible]

          DIMENSION IOVLAY(3,4)
          DATA IOVLAY(1,1),IOVLAY(2,1),IOVLAY(3,1)/'PA','SE','01'/
          DATA IOVLAY(1,2),IOVLAY(2,2),IOVLAY(3,2)/'PA',[illegible]
          DATA IOVLAY(1,3),IOVLAY(2,3),IOVLAY(3,3)/'PA',[illegible]
     C    DATA IOVLAY(1,4),IOVLAY(2,4),IOVLAY(3,4)/'PA',[illegible]
     C    DATA IOVLAY(1,5),IOVLAY(2,5),IOVLAY(3,5)/[illegible]

          CALL G[illegible](IPOL,[illegible],IATT,IERR)
          [illegible](1) = [illegible]
          IFLOW(1)=0
          IFLOW(2)=0
          CALL OVLAY(0,1,IOVLAY(1,1))

     10   CALL OVLAY(0,1,IOVLAY(1,IFLOW(1)))
          GO TO 10
          END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE ADDNOD(NODE)
C  SUBROUTINE ADDNOD ALLOWS FOR THE ADDITION OF NEW WAY POINTS.  A
C  MAXIMUM OF 10 WAY POINTS ARE ALLOWED AND A NODE CANNOT BE ADDED
C  BEYOND THE RECOVERY POINT.
C
C  NODE  -  THE NUMBER OF THE NODE TO PRECEED THE NEW NODE
C
      COMMON /ROUTE/ IPATH(65)
      IF (IPATH(1) .EQ. 10) GO TO 500

C  CALCULATE THE STARTING COORDINATES OF THE NEW NODE
      IPOS = 6*NODE-2
      IXNEW = FLOAT(IPATH(IPOS)+IPATH(IPOS+6))/2.
      IYNEW = FLOAT(IPATH(IPOS+1)+IPATH(IPOS+7))/2.

C  CHECK IF THE DISTANCE OF THE NEW NODE TO THE REFERENCE NODE IS WITHIN
C  10 UNITS.  IF TRUE, A NEW NODE WILL NOT BE ADDED AND A MESSAGE WILL APPEAR.
      IF (IABS(IXNEW-IPATH(IPOS)) .GT. 12) GO TO 50
      IF (IABS(IYNEW-IPATH(IPOS+1)) .LE. 12)GO TO 400
C
C  EXPAND IPATH ARRAY TO MAKE ROOM FOR THE NEW NODE
   50 JPOS = (IPATH(1)-NODE)*6+2
      IEND = 6*IPATH(1)+6
      DO 100 J=1,JPOS
         IPATH(IEND+6-J)=IPATH(IEND-J)
  100 CONTINUE
C
C  INCREASE THE TOTAL NUMBER OF NODES COUNTER AND (IF NECESSARY)
C  THE POSITION OF THE TARGET NODE
      IPATH(1) = IPATH(1) + 1
      IF (NODE .LT. IPATH(2)) IPATH(2) = IPATH(2)+1

C  FILL IN NEW NOD POSITION AND SET DEFAULT VALUES FOR AGL AND SPEED
      IPATH(IPOS+6) = IXNEW
      IPATH(IPOS+7) = IYNEW
      IPATH(IPOS+8) = IPATH(IPOS+2)
      IPATH(IPOS+9) = IPATH(IPOS+3)
      GO TO 510

C  TRYING TO ADD A NODE TOO CLOSE TO SURROUNDING NODES
  400 CALL ERMSG (2)
      GO TO 510

C  TRYING TO ADD MORE THAN 10 NODES.
  500 CALL ERMSG (1)
  510 CALL DRWPTH
      CALL GHLT
      CALL GCVT(0)
      CALL GSTT(0,0)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
        SUBROUTINE AWAIT
        COMMON /IATT/IATT(12),IERR

C       SUBROUTINE AWAITS FOR A DISPLAY INTERRUPT

        IATT(1)=0
        CALL CSTT(0,0)
1       CONTINUE
        IF(IATT(1).EQ.0)GO TO 1
        RETURN
        END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE BLAST
C  SUBROUTINE BLAST ALLOWS THE USER TO DESCRIBE THE TYPE AND
C AMOUNT OF STORES THE AIRCRAFT IS CARRYING.  A MENU OF THE
C TYPES OF STORES APPEARS AND A NUMBER PAD IS USED TO DESCRIBE
C THE AMOUNT.

      DIMENSION MAX(4)
      COMMON /MASTER/ MDF(2400)
C MAX IS GIVEN THE MAXIMUM POSSIBLE VALUE FOR NOW.  TO BE CHANGED
      DATA MAX /999,999,999,999/

      CALL CLIT (1,4,2)
      CALL CLIP (1,7,1)
      IPTR = MDF(9)
      NUMPTS = 4
      CALL CPTMNU (5)

C PUT CURRENT STORES VALUE INTO MENU
100   DO 120 K=0,3
         CALL UNIT (111+K)
         CALL GINT (21,MDF(IPTR+K),4)
120   CONTINUE
      CALL SELECT (NUMPTS,IOPT)
      IF (IOPT .EQ. 0) GO TO 150
      CALL CPTMNU(5)
      CALL CMSH (110+IOPT)
      CALL SPUNUM (MAX(IOPT),110+IOPT,24,NO)
      IF (NO .GE. 0) MDF(IPTR+IOPT-1) = NO
      GO TO 100

150   RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

PAGE 1 MFLEVI VORTXII FTN IV([illegible]) [illegible]

```
      SUBROUTINE [illegible]
      IMPLICIT INTEGER (A-Z)

C ***** FILLS CHARACTER AREAS AT LEFT SIDE OF SCREEN WITH BLANKS

      CALL GMLT

      DO 1 ENTITY=110,130
      CALL GSCH(ENTITY,6)
      WRITE(15,1000)
1000  FORMAT(' ')
1     CONTINUE

      CALL GSCH(110,35)
      WRITE(15,1000)
      CALL GS[illegible](0,0)

      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE CNGALT(NODE)
C  SUBROUTINE CNGALT ALLOWS THE USER TO CHANGE THE CURRENT
C  ALTITUDE OF A GIVEN LEG.
C  NODE - THE NODE NUMBER THAT IMMEDIATELY PRECEEDS THE  LEG TO
C  BE CHANGED.

      DIMENSION IALT(4)
      COMMON /ROUTE/ IPATH(65)
      DATA IALT /250,500,750,1000/

C  BLINK LEG  (IBLNK = ELEMENT # THAT BLINKS LEG)
      IBLNK = 6*NODE+3
      CALL GENT (300)
      CALL GPUT (IBLNK,130,3,1)
      [illegible]
      IF(NODE.GE.IPATH(1))IENT=IENT+1
      CALL GNIN (IENT)
        CALL GPUT (3,130,3,1)

C  DISPLAY OPTIONS, GET NEW SPEED, AND PUT INTO IPATH
      NHGHTS = 4
      CALL GITMU(4)
      CALL SELECT (NHGHTS,IOPT)
      IF (IOPT .EQ. 0) GO TO 100
      IPATH(6*NODE) = IALT(IOPT)

C  DISPLAY NEW VALUES AND RETURN
      CALL UPDATE
100   CALL GENT (300)
      CALL GPUT (IBLNK,130,3,0)
      CALL GNIN(IENT)
      CALL GPUT (3,130,3,0)

      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE CNGPTH
      COMMON /ROUTE/ IPATH(65)

C     THIS IS THE DRIVING ROUTINE TO ALTER A ROUTE

      DIMENSION ICNVRT(3)
      INTEGER TARNOD
      DATA ICNVRT/1,4,5/

C     WRITE WHEREABOUTS
      LCHNOD=1

10    CALL SELNOD(NODNUM)
C     NODNUM=0 IS USED TO EXIT THIS ROUTINE
      IF (NODNUM .EQ. 0) GO TO 100

      TARNOD=IPATH(2)
      MODE=1
      IF(NODNUM.EQ.LCHNOD.OR.NODNUM.EQ.TARNOD)MODE=2

C     WRITE OPTIONS IN THE MENU
      CALL CPTHMU(MODE)

C     IF THE LAUNCH NODE OR TARGET NODE IS CHODEN USE A DIFFERENT MENU
C     SELECT FROM MENU
      NMOPTS=5
      IF(MODE.EQ.2)NMOPTS=3
      CALL SELECT(NMOPTS,IOPT)
C     IOTP=0 INDICATES THAT THE RETURN KEY WAS HIT IN SELECT
      IF (IOPT .EQ. 0) GO TO 50
      [illegible]

50    IF(IOPT.EQ.3)CALL MOVNOD(NODNUM)
      CALL GROF (100)
      CALL TOCHEG (0)
      IF (IOPT.EQ.4)CALL CHGSPD(NODNUM)
      IF (IOPT.EQ.5)CALL CHGALT(NODNUM)
      CALL UNBLNK
      IF(IOPT.EQ.1)CALL ADDNOD(NODNUM)
      IF(IOPT.EQ.2)CALL DELNOD(NODNUM)
      GO TO 10

100   CALL GLMP (1,-1,0)
      CALL UNBLNK
      CALL TOFNSG (1)
      CALL GCYT (0)
      CALL GSTT (0,0)
      CALL GEOF(10)
      CALL GEOF(193)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE CHGSPD(NODE)
C  SUBROUTINE CHGSPD ALLOWS THE USER TO CHANGE THE CURRENT
C  SPEED OF A GIVEN LEG.
C  NODE - THE NODE NUMBER THAT IMMEDIATELY PRECEEDS THE LEG TO
C  BE CHANGED

      DIMENSION ISPD(4)
      COMMON /ROUTE/ IPATH(85)
      DATA ISPD /300,360,420,480/

C  BLINK LEG  (IBLNK = ELEMENT # THAT BLINKS LEG)
      IBLNK = 6*NODE+3
      CALL GENT (300)
      CALL GPUT (IBLNK,1[illegible]0,3,1)
      [illegible]
      I[illegible]IENT=IENT+1
      CALL [illegible]
      CALL GPUT (3,1[illegible],3,1)

C  DISPLAY OPTIONS, GET NEW ALTITUDE, AND PUT INTO IPATH
      NOPTS = 4
      CALL CTMENU(3)
      CALL SELECT (NOPTS,IOPT)
      IF (IOPT .EQ. 0) GO TO 100
      IPATH([illegible]NODE+1) = ISPD(IOPT)

C  DISPLAY NEW VALUES AND RETURN
      CALL [illegible]
100   CALL GENT (300)
      CALL GPUT (IBLNK,1[illegible]0,3,0)
      CALL [illegible]
      CALL GPUT (3,1[illegible],3,0)

      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE CNTOUR
      COMMON /DISP/IDPL(4000),MARKR(12)
      COMMON /MASTER/ MDF(2400)

C  ALLOW USER TO SELECT CONTOUR AND AGL
      IF (MARKR(1) .NE. 599) CALL GREST(IDPL,MARKR(2),MARKR(3),MARKR(4))
      CALL GSAVE (IDPL,MARKR(2),MARKR(3),MARKR(4))
      IENT = 599
      CALL TOPMSG(9)
      CALL CPTHMU(7)
      CALL SELECT (5,IWHCH)
      CALL CPTHMU(7)
      CALL GEON (IWHCH+110)
      CALL SELNUM (4,IWHCH+110,24,IAGL)
      IF (IAGL .LE. 0) GO TO 500

C  DISPLAY SENSOR CONTOURS
      IPTR = MDF(10)
      IF (IWHCH .EQ. 2 .OR. IWHCH .EQ. 5) GO TO 200
      IF (IWHCH .EQ. 4) GO TO 400
      NO = MDF(MDF(2))
      DO 120 J=1,NO
          NXTPTR = MDF(IPTR+IAGL +4*J-1)
          IF (NXTPTR .EQ. 0) GO TO 120
          CALL SENCNT (NXTPTR,IENT)
120   CONTINUE
      IF (IWHCH .NE. 3) GO TO 500
C  DISPLAY SAM AA CONTOURS
200   IPTR = MDF(3)
      IAGLPT = MDF(11)
      NPTS = MDF(IPTR[illegible])
      DO 220 J=1,NPTS
          [illegible]
          IX = [illegible]
          IY = [illegible]1)
          CALL [illegible],IX,IY)
          CALL [illegible]
          I[illegible]GL+4*J-5)
      CALL G[illegible]UT (6,1[illegible]),I[illegible]D,0)
220   CONTINUE
      IF (IWHCH .NE. 5) GO TO 500

C  DISPLAY COMPOSITE SENSOR CONTOUR
400   IPTR = MDF(10)
      NXTPTR = MDF(IPTR+IAGL-1)
      IF (NXTPTR .EQ. 0) GO TO 500
      CALL SENCNT (NXTPTR,IENT)
500   MARKR(1) = IENT
      CALL TOPMSG(1)
      CALL RCTMNU
      CALL GEOF (110+IWHCH)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE CPTHNU(MODE)
C     CHANGE PATH MENU
C     IF MODE=1 THEN LIST ALL 5 OPTIONS  IF MODE=2 THEN LIST
C     ONLY THE THREE OPTIONS AVAILABLE FOR LAUNCH AND TARGET NODES

      CALL GHLT
      GO TO (100,100,600,700,200,300,350),MODE

C  FOR MODES 1 AND 2 DISPLAY PATH OPTIONS
100   CALL GSCH(111,6)
      WRITE(15,111)
      IF(MODE.EQ.2)GO TO 500

      CALL GSCH(112,6)
      WRITE(15,112)
      CALL GSCH(113,6)
      WRITE(15,113)
      CALL GSCH(114,6)
      WRITE(15,114)
      CALL GSCH(115,6)
      WRITE(15,115)
111   FORMAT('ADD NODE')
112   FORMAT('DELETE NODE')
113   FORMAT('MOVE NODE')
114   FORMAT('CHANGE LEG SPEED')
115   FORMAT('CHANGE LEG ALTITUDE')
      GO TO 900

500   CALL GSCH(112,6)
      WRITE(15,114)
      CALL GSCH(113,6)
      WRITE(15,115)
      GO TO 900

C  FOR MODE 3 DISPLAY SPEED OPTIONS
600   CALL GSCH (111,6)
      WRITE (15,1600)
      CALL GSCH (112,6)
      WRITE (15,1601)
      CALL GSCH (113,6)
      WRITE (15,1602)
      CALL GSCH (114,6)
      WRITE (15,1603)
1600  FORMAT ('SPEED    I = 300 KNOTS')
1601  FORMAT ('          II = 360')
1602  FORMAT ('         III = 420')
1603  FORMAT ('          IV = 480')
      GO TO 900

C  FOR MODE 4 DISPLAY AGL OPTIONS
700   CALL GSCH (111,6)
      WRITE (15,1700)
      CALL GSCH (112,6)
      WRITE (15,1701)
      CALL GSCH (113,6)
      WRITE (15,1702)
      CALL GSCH (114,6)
      WRITE (15,1703)
```

```
1700  FORMAT ('  AGL   I = 250 FEET')
1701  FORMAT ('       II =  500')
1702  FORMAT ('      III =  750')
1703  FORMAT ('       IV = 1000')
      GO TO 900

C  MODE = 5  DISPLAY STORE OPTIONS
200   CALL GSCH (110,6)
      WRITE (15,1200)
      CALL GSCH (111,6)
      WRITE (15,1201)
      CALL GSCH (112,6)
      WRITE (15,1202)
      CALL GSCH (113,6)
      WRITE (15,1203)
      CALL GSCH (114,6)
      WRITE (15,1204)
1200  FORMAT ('SELECT STORES OPTION')
1201  FORMAT ('     500 LB BOMB')
1202  FORMAT ('    1000 LB BOMB')
1203  FORMAT ('    MISSILES')
1204  FORMAT ('    FUEL TANKS')
      GO TO 900

C  DISPLAY MAP MENU
300   CALL GSCH (110,6)
      WRITE (15,1300)
      CALL GSCH (111,6)
      WRITE (15,1301)
      CALL GSCH (112,6)
      WRITE (15,1302)
      CALL GSCH (113,6)
      WRITE (15,1303)
      CALL GSCH (114,6)
      WRITE (15,1304)
1300  FORMAT ('SELECT MAP')
1301  FORMAT ('     MAP 1')
1302  FORMAT ('     MAP 2')
1303  FORMAT ('     MAP 3')
1304  FORMAT ('     MAP 4')
      GO TO 900

C  CONTOURS MENU
350   CALL GSCH (110,6)
      WRITE (15,1350)
      CALL GSCH (111,6)
      WRITE (15,1351)
      CALL GSCH (112,6)
      WRITE (15,1352)
      CALL GSCH (113,6)
      WRITE (15,1353)
      CALL GSCH (114,6)
      [illegible]
      [illegible]
      [illegible]
1350  FORMAT ('SELECT CONTOUR OPTION')
1351  FORMAT ([illegible])
1352  FORMAT ([illegible])
```

AD-A101 860 INTEGRATED SCIENCES CORP SANTA MONICA CA F/G 15/7
CONCEPTS FOR HIGHLY INTERACTIVE AIR STRIKE PLANNING SYSTEMS (AS--ETC(U)
MAY 81 D H WALSH, G J REBANE, L R LEVI N00014-80-C-0812
UNCLASSIFIED ISC-330-2 NL

2 OF 2
AD A
101 860

END
DATE
FILMED
8-81
DTIC

```
117  1253  FORMAT (  [illegible]
118  [illegible]  FORMAT (  [illegible]
119  [illegible]  FORMAT (  [illegible]
120        [illegible]
121
122  9[illegible]  C[illegible] (0,0)
123        RETURN
124        END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DECLUT
      COMMON /FLAGS/ LMODE(9),RXPT(9),LSWCH(3),IFLOW(5)

C     LEVEL 1            TERRAIN        SFS        CONTOURS
C         LEVEL 2        TERRAIN        SFS
C         LEVEL 3        TERRAIN                   CONTOURS
C         LEVEL 4        TERRAIN

      [illegible]
      IF(L[illegible]-1
      [illegible]
      GO TO (100,200,300,400),LEVEL

100   CALL [illegible](1)
      CALL [illegible]
      GO TO [illegible]

200   CALL [illegible](1)
      CALL [illegible]
      GO TO [illegible]

300   CALL [illegible](0)
      CALL [illegible]
      GO TO [illegible]

400   CALL [illegible](0)
      CALL [illegible]
[illegible] RETURN
      END
0 [illegible] COMPILATION COMPLETE
```

```
      SUBROUTINE DIR1
      COMMON /FLAGS/ LMODE(9),MXPT(9),LSWCH(3),IFLOW(5)
      [illegible]
      INTEGER [illegible]

      IF (IFLOW(1).NE.0) GO TO 100
      CALL [illegible]
      CALL [illegible]

  100 CALL [illegible]
      IF (IFLOW(1).NE.1) GO TO 900
      [illegible]
      IF [illegible] CALL RECALL
      IF [illegible] CALL SPACE
      IF [illegible] CALL EXIT
      IF [illegible] CALL [illegible]

      IFLOW (1) = 1
      IFLOW (2) = 0
      GO TO 100

  900 RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE [illegible]
      COMMON [illegible] LIMIT(9),[illegible](9),[illegible](9),IFLOW([illegible])
      LOGICAL [illegible]
      INTEGER [illegible]

      [illegible]=IFLOW(2)

      IF([illegible].EQ.2)CALL [illegible]
      IF([illegible].EQ.2)CALL [illegible]
      IFLOW(1)=1
      IFLOW(2)=0
      RETURN
      END
 0 ERRORS [illegible]
```

```
      SUBROUTINE [illegible]
      COMMON [illegible](2),[illegible]T(2),L[illegible]H(2),IFLOW(2)
      LOGICAL [illegible]
      INTEGER OPTION

      OPTION = IFLOW(2)
      IF([illegible].EQ.[illegible]) [illegible]
      IF([illegible].EQ.[illegible]) [illegible]
      IF([illegible].EQ.[illegible]) [illegible]
      IF([illegible].EQ.[illegible]) [illegible]
      IFLOW(1) = 1
      IFLOW(2) = 0
      RETURN
      END
 0 ERRORS COMPILATION [illegible]
/END

/ASSIGN,SI,CP
```

```
      SUBROUTINE DRECTR

C     THIS ROUTINE INFORMS THE USER OF THE OPTIONS AVAILABLE
C     BY TURNING ON THE LIGHTS ON THE KEYS.  WHEN AN OPTION HAS BEEN
C     SELECTED IFLOW(1) IS SET TO THE OVERLAY NUMBER  AND IFLOW(2)
C     IS SET TO THE ROUTINE NUMBER

      IMPLICIT INTEGER(A-Z)
      COMMON /MASTER/ MDF(2400)
      COMMON /FLAGS/ LMODE(9),RXPT(9),LSWCH(3),IFLOW(5)
      REAL RXPT
      LOGICAL LSWCH
      LOGICAL NOPATH

      CALL TOPMSG(13)
      VLUATE=4
      OPTIMZ=5
      STORE=11
      DCLTER=12
      CNTOUR=13
      CNGPTH=20
      GETSPS=21
      GETLTP=28
      RECALL=29
      EXIT=31

      NOPATH=.FALSE.
      IF(MDF(MDF(8)).EQ.0)NOPATH=.TRUE.

      IF(NOPATH)CALL LAMPS(176,32,56,0)
      IF(.NOT.NOPATH)CALL LAMPS(176,48,56,48)

 10   CALL CKINT(KEY)
      IF(KEY.EQ.-1)GO TO 10

      IF(NOPATH)GO TO 130
 110  IF(KEY.NE.VLUATE)GO TO 120
          IFLOW(1)=3
          IFLOW(2)=6
           GO TO 900

 120  IF(KEY.NE.OPTIMZ)GO TO 130
      IF(NOPATH)GO TO 130
          IFLOW(1)=3
          IFLOW(2)=5
          GO TO 900

 130  IF(KEY.NE.STORE)GO TO 140
          IFLOW(1)=1
          IFLOW(2)=2
          GO TO 900

 140  IF(KEY.NE.DCLTER)GO TO 150
          IFLOW(1)=1
          IFLOW(2)=6
          GO TO 900

 150  IF(KEY.NE.CNTOUR)GO TO 160
```

```
              IFLOW(1)=3
              IFLOW(2)=4
              GO TO 900

 160   IF(KEY.NE.CNGPTH)GO TO 170
       IF(NOPATH)GO TO 170
              IFLOW(1)=2
              IFLOW(2)=2
              GO TO 900

 170   IF(KEY.NE.GETSPS)GO TO 180
              IFLOW(1)=2
              IFLOW(2)=3
              GO TO 900

 180   IF([illegible])GO TO [illegible]
              IFLOW(1)=3
              IFLOW(2)=1
              GO TO 900

 200   IF(KEY.NE.[illegible])GO TO 210
              IFLOW(1)=1
              IFLOW(2)=1
              GO TO 900

 210   IF(KEY.NE.[illegible])GO TO 220
              IFLOW(1)=1
              IFLOW(2)=5
              GO TO 900

 220   GO TO 10

 900   CALL [illegible](1,-1,0)
       CALL [illegible]
       RETURN
       END
O ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DRWPTH
C  SUBROUTINE DRWPTH DRAWS THE PATH IN THE COMMON AREA /ROUTE/
C  IT EXPECTS TO FIND AN ARRAY OF LENGTH 55 CALLED IPATH IN THE
C  COMMON AREA.  THE  LAUNCH, TARGET, AND TERMINAL NODES WILL BE
C  COLORED DIFFERENTLY THAN THE OTHER NODES.

      COMMON /ROUTE/ IPATH(55)
      COMMON /FLAGS/ LMODE(9),RXPT(9),LSWCH(3),IFLOW(5)

C  PRINT CIRCLE FOR LAUNCH NODE
      L1=IPATH(1)
      L2 = IPATH(2)-1
      IEL = 7
      ICOL = 1
      CALL GENT(300)
      CALL GPUT(1,100,IPATH(4),0)
      CALL GPUT(2,110,IPATH(5),0)
      CALL GPUT (5,1700,0,0)
      CALL GPUT (6,140,6,0)

C  ON FIRST PASS, SET UP ENTITY ONLY
      IF (L1 .NE. 0) GO TO 90
          ICNT = 1
          CALL GPUT (7,1774,255,0)
          GO TO 105

 90   CALL GPUT (7,1000,10,0)
C  DISPLAY LEGS AND CIRCLES FOR REMAINING NODES
      DO 100 ICNT =1,L1
          J = 6*ICNT+4
         CALL GPUT (NXT(IEL),140,6,ICOL)
         CALL GPUT (NXT(IEL),130,3,0)
         CALL GPUT (NXT(IEL),53,IPATH(J),IPATH(J+1))
         IF ((ICNT .EQ. 1) .OR. ICNT .EQ. L1) ICOL = 0
         CALL GPUT (NXT(IEL),140,6,ICOL)
         CALL GPUT (NXT(IEL),130,3,0)
         CALL GPUT (NXT(IEL),1000,10,0)
         ICOL = 1
  100 CONTINUE

C  FILL IN NO DATA TO LEAVE SPACE IN IDPL ARRAY FOR POSSIBLE
C  FUTURE ADDED NODES
      IF (ICNT .GT.10) RETURN
 105  DO 110 J=ICNT,10
         CALL GPUT (NXT(IEL),1774,255,0)
         CALL GPUT (NXT(IEL),1774,255,0)
         CALL GPUT (NXT(IEL),73,0,0)
         CALL GPUT (NXT(IEL),1774,255,0)
         CALL GPUT (NXT(IEL),1774,255,0)
         CALL GPUT (NXT(IEL),1774,255,0)
  110 CONTINUE

          IF(L1.NE.0.AND.LMODE(2).NE.1)CALL LEGTAB
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DELNOD (NODE)
C  SUBROUTINE DELNOD ALLOWS FOR THE DELETION OF A NODE.  THE TARGET
C  LAUNCH, OR RECOVERY NODES CANNOT BE DELETED
C
C    NODE - [illegible] TO BE DELETED
C
      COMMON /[illegible]/
      IF (NODE .EQ. 1 .OR. NODE .EQ. IPATH(2)) RETURN
C
C  COMPRESS THE IPATH ARRAY
      [illegible]
      [illegible]
      DO 100 J [illegible]
         [illegible]
  100 CONTINUE
C
C  RESET THE NUMBER OF NODES AND THE TARGET NODE (IF NECESSARY)
      IPATH(1) = IPATH(1) - 1
      IF (NODE .LT. IPATH(2)) IPATH(2)=IPATH(2)-1
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      RETURN
      END
0 ERRORS [illegible]
```

```
      SUBROUTINE [illegible] (IENT,IX,IY)
C   THIS ROUTINE DRAWS THE [illegible] SIDE.
C
      CALL GETS (IENT,IX,IY)
      CALL [illegible] (2)
      CALL CHUT (6,[illegible],-7,-10)
      CALL CHUT (7,[illegible],14,0)
      CALL CHUT ([illegible],10,-7,10)
      CALL CHUT (9,[illegible],8,0)
      CALL CHUT (10,[illegible])
      CALL CHUT (11,[illegible],0)
      CALL CHUT (12,[illegible])
      CALL CHUT (13,[illegible],0)
      CALL CHUT (14,10,[illegible])
      RETURN
      END
```

0 ERRORS COMPILATION COMPLETE

```
      SUBROUTINE [illegible](IGHT,I[illegible],IY)
C     THIS ROUTINE [illegible]

      CALL [illegible](IGHT,I[illegible],[illegible])
      CALL [illegible]
      CALL [illegible](6,[illegible],[illegible],0)
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible](10,[illegible],[illegible],-7)
      CALL [illegible](11,[illegible],[illegible],0)
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible](16,[illegible],10,0)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DRWSPS
C  THIS SUBROUTINE TURNS ON SIGNIFICANT POINTS TO BE DISPLAYED

      COMMON /MASTER/ MDF(2400)

C  DISPLAY RADAR SIGNIFICANT POINTS
      IPTR = MDF(6)
      IEND = MDF(IPTR)
      IF (IEND .EQ. 0) GO TO 30
      DO 20 J=1,IEND
         [illegible]
         CALL PTURN (J)
         CALL PLOT (1,1,0,MDF(IPTR+IPOS),0)
         CALL PLOT (2,110,MDF(IPTR+IPOS+1),0)
         CALL PTURN (J)
   20 CONTINUE

C  DISPLAY VISUAL SIGNIFICANT POINTS
   30 IPTR = MDF(7)
      IEND = MDF(IPTR)
      IF (IEND .EQ. 0) GO TO 50
      DO 40 J=1,IEND
         IPOS = 2*J-1
         CALL PTURN (J)
         CALL PLOT (1,1,0,MDF(IPTR+IPOS),0)
         CALL PLOT (2,110,MDF(IPTR+IPOS+1),0)
         CALL PTURN (J)
   40 CONTINUE

   50 RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DSPSCN
C  SUBROUTINE DSPSCN DISPLAYS THE SCENARIO, RADAR, AND SAM SITES.  IT
C  EXPECTS TO FIND THE POINTS FOR THE TERRAIN AND THE COORDINATES FOR
C  THE SAM AND RADAR SITES IN AN ARRAY MDF WHICH IS LOCATED IN THE
C  COMMON AREA MASTER. THE SCENARIO WILL BE FOUND IN ENTITIES 200 TO 299.
C  SEE DOCUMENTATION AS TO THE FORM OF THE MDF ARRAY.

      COMMON /MASTER/ MDF(2400)
      IENT = 199

C  DRAW IN ORDER: (1) TERRAIN, (2) RADAR SITES,  (3) SAM SITES
10    DO 100 J=1,3
         IPTR = MDF(J)
         N = MDF(IPTR)
         IF (N.EQ.0) GO TO 100
         IPTR = IPTR+1
      IF (J.GT.1) GO TO 75

C  DISPLAY [illegible] FEATURES.
         DO [illegible]
               CALL [illegible],MDF,IPTR+4,MDF(IPTR+[illegible]
               CALL [illegible](IPTR),IENT)
               [illegible](IPTR+3)
50       CONTINUE
         GO TO 100

C  DRAW THE [illegible] SITES
75          IF [illegible](IENT),MDF(IPTR),[illegible]
            IF [illegible](IENT),MDF(IPTR),[illegible]
            IF [illegible] TO 100
            [illegible]
            DO [illegible]
            IPTR =IPTR+[illegible]
               CALL [illegible](IENT),NOLD,MDF(IPTR),MDF(IPTR+1)
            CALL [illegible]
90          CONTINUE
100   CONTINUE
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE ERRMSG(NO)
      COMMON /ATT/ IATT(12),IERR
C  SUBROUTINE ERRMSG PRINTS ALL ERROR MESSAGES BELOW THE SCENARIO.  THIS
C  ROUTINE EXPECTS ENTITY 191 TO HAVE BEEN SET UP TO ALLOW FOUR LINES OF
C  EACH LINE WITH A MAXIMUM OF 50 CHARACTERS.  AFTER THE MESSAGE IS DISPLAYED,
C  THE ROUTINE SITS IN A LOOP UNTIL THE RETURN KEY IS PRESSED.

      CALL GLMP (1,-1,0)
      CALL GLMP (1,7,1)

C  BLINK ''ERROR'' MESSAGE
      CALL GENT(191)
       CALL GPUT (3,130,3,1)
      CALL GPUT (167,140,6,0)
      CALL GSCH (191,6)
      WRITE (15,500)
  500 FORMAT (20X,'-- ERROR --')
      GO TO (1,2,3),NO

    1 CALL GSCH (191,60)
      WRITE (15,10)
   10 FORMAT (5X,'THE MAXIMUM NUMBER OF ALLOWABLE WAY POINTS')
      CALL GSCH(191,114)
      WRITE (15,11)
   11 FORMAT (5X,'-10- HAS BEEN EXCEEDED!')
      GO TO 505

    2 CALL GSCH (191,60)
      WRITE (15,20)
   20 FORMAT (5X,'WAY POINT NOT ADDED SINCE IT LIES TOO ')
      CALL GSCH(191,114)
      WRITE (15,21)
   21 FORMAT (5X,'CLOSE TO THE SURROUNDING WAY POINTS!')
      GO TO 505

C  NUMBER PAD ROUTINE ERROR.  NUMBER ENTERED GREATER THAN MAX
    3 [illegible]
      WRITE (15,[illegible])
      CALL GSCH (191,114)
      WRITE (15,3005)
 3000 FORMAT ('  INPUTTED NUMBER IS GREATER THAN THE MAXIMUM.')
 3005 FORMAT ('  CHECK ON THE RANGE AND INPUT THE NUMBER AGAIN')
      GO TO 505

  505 CALL GSCH (191,168)
      WRITE (15,510)
  510 FORMAT (10X,'PRESS ''RETURN'' TO CONTINUE . . .')
      CALL GEON (191)

C  WAIT FOR [illegible] INTERRUPT TO CONTINUE WITH THE PROGRAM.
      [illegible]
      CALL [illegible],0)
  610 CALL [illegible]
      IF [illegible] GO TO 610
      CALL [illegible] (1,[illegible],0)
      IATT([illegible]) = 0
      CALL [illegible],0)
      CALL [illegible],3,0)
```

```
      CALL [illegible] (2,1[illegible],0,0)
      CALL [illegible] (1[illegible],1[illegible],0,1)
      CALL [illegible] ([illegible])
      RETURN
      END
 0 ERRORS [illegible]
```

```
      SUBROUTINE [illegible]
      DIMENSION [illegible](120)
      COMMON [illegible](120)
      DATA [illegible],[illegible],[illegible](10) /2HOD,2H[illegible],2H[illegible]/

      [illegible] = 0
      [illegible] = 0
      CALL [illegible] (21,[illegible],0)
      WRITE (21) ([illegible](J),J=1,120)
      CALL [illegible] (21,0)
      STOP
      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE GETLTP
C  SUBROUTINE GETLTP ALLOWS THE USER TO SPECIFY THE POSITION OF THE LAUNCH,
C TARGET, AND RECOVERY NODES.  A CURSOR APPEARS DESCRIBED BY ENTITY 10.
C  THE USER MOVES THE CURSOR TO THE DESIRED POSITION AND PRESSES THE
C  ACCEPT KEY TO SPECIFY A NODE POSITION.

      COMMON /ATT/ IATT(12),IERR
      COMMON /ROUTE/ IPATH(65)
      COMMON /MASTER/ MDF(2400)
      COMMON /FLAGS/ LMODE(9),RXPT(9),LSWCH(3),IFLOW(5)

C     TURN OFF OLD ROUTE
      CALL GEOF(300)
      CALL GEOF(310)
      CALL GEOF(311)
      CALL GEOF(312)
      DO 3 J=1,11
          CALL GEOF (J+149)
 3    CONTINUE
      DO 4 J=1,4
      CALL GENT(110+J)
      CALL GPUT (23,90,1H ,0)
      CALL GPUT (24,90,1H ,0)
      CALL GPUT (25,90,1H0,0)
      MDF(MDF(9)-1+J) = 0
 4    CONTINUE

C  INITIALIZE PATH ARRAY AND TURN ON CYCLE TIMER
      ISWCH = 0
      IPTR = MDF(8)
      IPOS = 0
      IPATH (1) = 2
      IPATH (2) = 2
      CALL TOPMSG(6)
      CALL GMLT
      CALL GSTT(1)

C  SET CURSOR TO CURRENT TRACKBALL POSITION
      CALL GTRACK (IX,IY)
      CALL GENT(10)
      CALL GPUT (1,100,IX,0)
      CALL GPUT (2,110,IY,0)
      CALL GEON (10)

C  GET IN ORDER LAUNCH, TARGET, AND RECOVERY NODES
      DO 100 J = 4,16,6
          K = FLOAT(J+2)/6.+309.
          CALL MESSAG (K-306)
          LSWCH(1) = .FALSE.
          CALL GEOF (100)
          CALL TOPMSG(2)
          CALL LAMPS (0,3,16,130)
 5    IATT(1) = 0
      CALL GSTT(0,0)

C  CHECK ON INTERRUPTS
 10       CALL CKINT(KEY)
          IF (KEY .EQ. -1) GO TO 10
```

```
          IF (KEY .EQ. 1) GO TO 50
          IF (KEY .EQ. 19) GO TO 75
          IF (KEY .EQ. 7) GO TO 200
          IF (KEY .EQ. 12) CALL DECLUT
          IF (KEY .NE. 40) GO TO 5

C   CYCLE TIMER INTERRUPT!  MOVE CURSOR TO CURRENT POSITION
      CALL GTRACK (IX,IY)
          ISWCH = 1-ISWCH
          CALL GENT (10)
          CALL GPUT (1,100,IX,0)
          CALL GPUT (2,110,IY,0)
          IF (ISWCH .EQ. 0) GO TO 30
          CALL LATLON (IX,IY)
          GO TO 10

C   RANGE AND BEARING UPDATE
 30       IF (.NOT. LSWCH(1)) GO TO 10
          CALL RANGE (IX,IY)
          CALL LAMPS (0,0,0,130)
          GO TO 10

C   KEYBOARD INTERRUPT!
C   SET APPROPRIATE ELEMENTS IN IPATH ARRAY TO LAUNCH, TARGET, RECOVERY POSITIONS
 50       IPATH(J) = IX
          IPATH(J+1) = IY
          MDF(IPTR+IPOS)=IX
          MDF(IPTR+IPOS+1) = IY
          IPOS = IPOS+2

C   SET ON FIGURE OVER THE APPROPRIATE NODE.  K GETS THE ENTITY NUMBER.
          CALL CENT (K)
          CALL GPUT (1,100,IX,0)
          CALL GPUT (2,110,IY,0)
          CALL GEON (K)
          GO TO 100

C   RANGE AND BEARING ROUTINE - GET REFERENCE POINT
 75       IF (LSWCH(1)) GO TO 30
          IX=-100
          IY = 0
          CALL RANGE (IX,IY)
          CALL MESSAG (K-306)
          CALL GLMP (1,1,1)
          IF  (.NOT.LSWCH(1)) CALL GLMP (1,19,1)
          GO TO 10
 100  CONTINUE

C   SET AGL AND SPEED DEFAULT VALUES
      IPATH(6)=500
      IPATH(7)=360
      IPATH(12) = 500
      IPATH(13) = 360
      CALL DPWTH
      CALL GEON(300)

C     TURN OFF LIGHTS, HALT CYCLE TIMER THEN RETURN
 150  CALL GEOF(10)
```

```
      CALL [illegible]
      CALL [illegible]
      IF [illegible] CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]1,-1,0)
      CALL [illegible]
      CALL [illegible]
      [illegible]
      CALL [illegible](0,0)
      RETURN

C     [illegible] TABLE
  200 CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      GO TO [illegible]
      END
0 ERRORS COMPILATION COMPLETE.
/EOI
/ASSIGN,SI,CR
```

```
      SUBROUTINE GETSPS
C  SUBROUTINE GETSP ALLOWS THE USER TO IDENTIFY RADAR SIGNIFICANT
C  POINTS AND VISUALLY S.P.  A MAXIMUM OF 10 EACH IS ALLOWED.

      COMMON /ATT/ IATT(12),IERR
      COMMON /MASTER/ MDF(2400)
      COMMON /FLAGS/ LMODE(9),EXPT(9),LSWCH(3),IFLOW(5)

C     TURN OFF OLD SPS
      DO 3 IENT=370,389
        CALL GEOF(IENT)
3     CONTINUE

C  INITIALIZE
      ISWCH = 0
      IPTR=MDF(6)
      CALL GHLT
      CALL GCYT(1)
      IENT = 369
      CALL TOFMSG(5)

C  SET CURSOR TO CURRENT TRACKBALL POSITION.
      CALL GENT(10)
      CALL GTRACK (IX,IY)
      CALL GPUT (1,100,IX,0)
      CALL GPUT (2,110,IY,0)
      CALL GEON (10)

C  GET UP TO A MAXIMUM OF 10 RADAR AND VISUALLY SIGNIFICANT POINTS
      DO 200 K=1,2
          MDF (IPTR) = 0
          CALL MESSAG (K+6)
      DO 100 J=1,10
          LSWCH(1) = .FALSE.
          CALL GEOF (130)
          CALL TOFMSG(0)
          CALL [illegible] (?,0,130)
5         IATT(1) = 0
          CALL GSTT(0,0)

C  CHECK ON INTERRUPTS
10        CALL CKINT(KEY)
              IF (KEY .EQ.-1) GO TO 10
              IF (KEY .EQ.7) GO TO 150
              IF (KEY .EQ.1) GO TO 50
              IF (KEY .EQ. 19) GO TO 40
              IF (KEY .NE. 40) GO TO 5

C  CYCLE TIMER INTERRUPT. DISPLAY CURRENT TRACKBALL POSITION.
          CALL GTRACK (IX,IY)
          ISWCH = 1-ISWCH
          CALL GENT (10)
          CALL GPUT (1,100,IX,0)
          CALL GPUT (2,110,IY,0)
          IF (ISWCH .EQ. 0) GO TO 25
          CALL LATLON (IX,IY)
          GO TO 10
```

```
C  UPDATE RANGE AND BEARING
25         IF (.NOT.LSWCH(1)) GO TO 10
           CALL RANGE (IX,IY)
           CALL LAMPS (0,0,0,130)
           GO TO 10

C  RANGE AND BEARING ROUTINE
40         IF (LSWCH(1)) GO TO 25
           IX=-100
           IY = 0
           CALL RANGE (IX,IY)
           CALL MESSAG (K+6)
           CALL LAMPS (0,0,0,130)
           IF (.NOT. LSWCH(1)) CALL GLMP (1,19,1)
           GO TO 10

C  ACCEPT KEY INTERRUPT.  PLACE SIGNIFICANT POINTS IN MDF
C  AND DRAW SYMBOL IN PROPER LICATION ON THE SCREEN.
50         MDF(IPTR+2*J-1) = IX
           MDF(IPTR+2*J)   = IY

C  TURN ON SIGNIFICANT POINT ENTITY
           CALL [illegible]
           [illegible]
           [illegible]
           [illegible]
100    CONTINUE
150        [illegible]
           [illegible]
200    CONTINUE

C  [illegible] TO [illegible] TO CALLING PROGRAM
       CALL [illegible] (1,-1,[illegible])
       CALL [illegible]
       CALL [illegible]
       CALL [illegible]
       CALL [illegible]
       CALL [illegible](10)
       CALL GHLT
       CALL [illegible](0)
       INTT(1) = 0
       CALL GSTT(0,0)
       RETURN
       END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE [illegible]
C  SUBROUTINE [illegible] THE CURRENT POSITION OF THE [illegible]
C  AND [illegible] THE VISIBLE POSITION OF THE [illegible]
C     IX -  THE [illegible] POSITION
C     IY -  THE [illegible] Y POSITION
C  THE [illegible]
C          X-COORDINATE      (-[illegible])
C          Y-COORDINATE      (-[illegible])
C
      IDEV=1
      CALL [illegible]
      IX = [illegible]
      IY = [illegible]
      IF [illegible]
      IF [illegible]
      IF [illegible]
      IF [illegible]
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE TITLE
      [illegible]
      [illegible]
      [illegible]
      [illegible] (LIGHT,TEMP)
C     THIS WILL [illegible] FOR DISPLAY 1
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      RETURN
      END
0 ERRORS [illegible]
```

```
      SUBROUTINE INSEN (MAP,IVERS)
C  SUBROUTINE INSEN READS FROM THE DISK THE SCENARIO INTO THE
C  MASTER DATA FILE.
      COMMON /MASTER/ MDF(2400)
      INTEGER IFCB(13),SCENE(3,24)
      DATA SCENE(1,1),SCENE(2,1),SCENE(3,1) /2HSC,2HEN,2H11/
      DATA SCENE(1,2),SCENE(2,2),SCENE(3,2) /2HSC,2HEN,2H21/
      DATA SCENE(1,3),SCENE(2,3),SCENE(3,3) /2HSC,2HEN,2H31/
      DATA SCENE(1,4),SCENE(2,4),SCENE(3,4) /2HSC,2HEN,2H41/
      DATA SCENE(1,5),SCENE(2,5),SCENE(3,5) /2HSC,2HEN,2H12/
      DATA SCENE(1,6),SCENE(2,6),SCENE(3,6) /2HSC,2HEN,[illegible]/
      DATA SCENE(1,7),SCENE(2,7),SCENE(3,7) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1,8),SCENE(2,8),SCENE(3,8) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1,9),SCENE(2,9),SCENE(3,9) /2HSC,2HEN,2H13/
      DATA SCENE(1,10),SCENE(2,10),SCENE(3,10) /2HSC,2HEN,2H23/
      DATA SCENE(1,11),SCENE(2,11),SCENE(3,11) /2HSC,2HEN,2H33/
      DATA SCENE(1,12),SCENE(2,12),SCENE(3,12) /2HSC,2HEN,2H43/
      DATA SCENE(1,13),SCENE(2,13),SCENE(3,13) /2HSC,2HEN,2H14/
      DATA SCENE(1,14),SCENE(2,14),SCENE(3,14) /2HSC,2HEN,2H24/
      DATA SCENE(1,15),SCENE(2,15),SCENE(3,15) /2HSC,2HEN,2H34/
      DATA SCENE(1,16),SCENE(2,16),SCENE(3,16) /2HSC,2HEN,2H44/
      DATA SCENE(1,17),SCENE(2,17),SCENE(3,17) /2HSC,2HEN,2H15/
      DATA SCENE(1,18),SCENE(2,18),SCENE(3,18) /2HSC,2HEN,2H25/
      DATA SCENE(1,19),SCENE(2,19),SCENE(3,19) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1,20),SCENE(2,20),SCENE(3,20) /2HSC,2HEN,2H45/
      DATA SCENE(1,21),SCENE(2,21),SCENE(3,21) [illegible]
      DATA [illegible](1,22),[illegible](2,22),[illegible](3,22) [illegible]
      [illegible]
      [illegible]

      IFCB(2) = 0
      IFCB(3) = 0
      IF [illegible] .NE. 0) GO TO 25
      INNCH = [illegible]
      GO TO 50
 25   IF (IVERS .EQ. 1) RETURN
      INNCH = (MAP+IVERS-1)+1

 50   IFCB(8) = SCENE(1,INNCH)
      IFCB(9) = SCENE(2,INNCH)
      IFCB (10) = SCENE(3,INNCH)

      CALL VEOPEN (21,12,IFCB,0)
      READ (21) (MDF(J),J=1,2400)
      CALL VICLOS (21,0)
      RETURN
      END
```

0 ERRORS COMPILATION COMPLETE

```
      SUBROUTINE LATLON (IX,IY)
C  SUBROUTINE LATLON CONVERTS THE CURRENT POSITION OF THE
C  CURSOR FROM MINUTES TO DEGREES LATITUDE AND MINUTES AND
C  DEGREES LONGITUDE AND MINUTES.  THE POSITION IS DISPLAYED
C  ON THE SCREEN IN ENTITY NUMBER 193.
C  THE 4TH ELEMENT OF THE MDF ARRAY POINTS TO THE ORIGIN
C  LAT AND LONG POSITION EXPRESSED IN MINUTES AND THE
CC WIDTH OF THE SCENARIO IN KILOMETERS.

      COMMON /MASTER/ MDF(2400)
      CALL CENH (193)
      CALL CENT (193)
      IPTR = MDF(5)
      CVFACT = FLOAT(MDF(IPTR+2))/1024.

C  GET CURRENT LATITUDE
      [illegible] = [illegible] (CVFACT*FLOAT(IY)+.5)
      [illegible]
      L[illegible] = [illegible]
      IF [illegible] LAT=100-LAT

C  GET CURRENT LONGITUDE [illegible] LATITUDE
C  [illegible] LONGITUDE [illegible] LATITUDE)
C  [illegible]
      [illegible] (FLOAT(L[illegible])*FLOAT(L[illegible]
      [illegible] (FLOAT(I[illegible]) CVFACT/CVFLO[illegible]
      LON = [illegible]
      [illegible]
      IF [illegible] = 100-LON

C  DISPLAY [illegible] LATITUDE AND LONGITUDE ON THE SCREEN
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE LEGTAB
C  SUBROUTINE LEGTAB DISPLAYS FOR EACH LEG, THEIR RESPECTIVE
C  SPEED AND ALTITUDE.  THE COLOR IN THE TABLE IS TO MATCH
C  THE LEG COLOR TO IDENTIFY THE SAFETY OF THE PATH.

      COMMON /ROUTE/ IPATH(65)

      CALL GEOM (150)
      N= IPATH(1)
      NP1=N+1
      ITAR=IPATH(2)

C  PUT IN PATH SPEED AND ABOVE GROUND LEVEL (AGL)

      LEGNUM=1

      DO 100 J=1,NP1
         IENT=15*J+[illegible]
         [illegible]
            CALL [illegible]
            [illegible]
            GO TO 100
 
75       CALL [illegible]
         [illegible]
         [illegible]
         CALL [illegible],4)
         [illegible],5)
         [illegible]
         [illegible]
100   CONTINUE

C  BLANK OUT TABLE [illegible] NOT IN USE
      [illegible]
      DO 120 [illegible]
         CALL [illegible]
120   CONTINUE
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE LINTYP (NO,IENT)
C  THIS SUBROUTINE DRAWS FOR A GIVEN ENTITY ITS LINES IN
C  VARIOUS COLORS AND FORMS.  CONTROL REGISTER (ELEMENT 3)
C  CONTROLS THE LINE TYPE.  DISPLAY REGISTER (ELEMENT 4) CONTROLS COLOR.
      DATA IRED,IYELLW,IORANG,IGREEN /0,1,2,3/

      CALL GENT (IENT)
      GO TO (1,2,3,4,5,6,7,8,9,10),NO
C  LINE TYPE 1 - RED DASHED LINE
1     CALL COLOR (IRED)
      CALL GPUT (3,130,1,2)
      RETURN
C  TERRAIN TYPE 2 - YELLOW DASHED LINE
2     CALL COLOR (IYELLW)
      CALL GPUT (3,130,1,2)
      RETURN
C  LINE TYPE 3 - ORANGE DASHED LINE
3     CALL COLOR (IORANG)
      CALL GPUT (3,130,1,2)
      RETURN
C  LINE TYPE 4 - ORANGE DOTTED LINE
4     CALL COLOR (IORANG)
      CALL GPUT (3,130,1,1)
      RETURN
C  LINE TYPE 5 - GREEN DOTTED LINE
5     CALL COLOR (IGREEN)
      CALL GPUT (3,130,1,1)
      [illegible]
C  LINE [illegible] LINE
6     CALL COLOR [illegible]
      CALL [illegible] 1,1)
      RETURN
C  LINE TYPE 7 - [illegible]-DOTTED LINE
7     CALL COLOR [illegible]
      CALL GPUT (3,1[illegible],1,3)
      RETURN
C  LINE TYPE 8 - YELLOW DASH-DOT
8     CALL COLOR (IYELLW)
      CALL GPUT (3,130,1,3)
      RETURN
C  LINE TYPE 9 - GREEN DASH-DOT
9     CALL COLOR (IGREEN)
      CALL GPUT (3,130,1,3)
      RETURN
C  LINE TYPE 10 - YELLOW SOLID
10    CALL COLOR (IYELLW)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      [illegible],IOPT,IUNIT,[illegible])
      [illegible]
C     [illegible]
C     [illegible]
C     [illegible]
C     [illegible]
C     [illegible]
C     [illegible] POINTS TO PLOT
      CALL [illegible](NPTS+1))
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
         CALL [illegible](J+1))
         [illegible]
  110 CONTINUE
      RETURN
      END
0 ERRORS [illegible]
```

```
      SUBROUTINE [illegible]
      [illegible]
C     [illegible] ON THE LEFT SIDE OF [illegible]
      DO [illegible]
      [illegible]
    1 CONTINUE
      RETURN
      END
0 ERRORS [illegible] COMPILATION COMPLETE [illegible]
```

```
      SUBROUTINE [illegible]
      [illegible]

C     [illegible] AT THE LEFT SIDE OF THE [illegible]

      DO 1 [illegible]
      CALL [illegible]
    1 CONTINUE

      RETURN
      END
0 [illegible] COMPILATION [illegible]
```

```
      SUBROUTINE MESSAG(NO)
C  SUBROUTINE MESSAG DISPLAYS BELOW THE SCENARIO BORDER A MESSAGE
C  DESCRIBING AN INSTRUCTION OR AN ERROR MESSAGE.  THE AREA USED FOR
C  THE MESSAGES IS SET UP IN ENTITY 191.

C  BLANK OUT MESSAGE AREAS
      CALL GSCH (191,6)
      WRITE (15,1110)
      CALL GSCH (191,60)
      WRITE (15,1110)
      CALL GSCH (191,114)
      WRITE (15,1110)
      CALL GSCH (191,168)
      WRITE (15,1110)
1110  FORMAT ('          ')
      GO TO (100,200,300,350,400,450,500,550,600,650),NO

C  MOVE NODE MESSAGE
100   CALL GSCH (191,6)
      WRITE (15,110)
110   FORMAT (15X,'--MOVE NODE--')
      CALL GSCH(191,60)
      WRITE (15,115)
115   FORMAT (' MOVE NODE TO DESIRED POSITION AND PRESS ''ACCEPT''.')
      CALL GSCH (191,114)
      WRITE (15,120)
120   FORMAT(' ''RETURN'' KEY REPOSITIONS NODE TO INITIAL LOCATION.')
      GO TO 999

C  SELECT NODE MESSAGE
200   CALL GSCH(191,6)
      WRITE (15,210)
210   FORMAT(14X,'--SELECT NODE--')
      CALL GSCH (191,60)
      WRITE (15,215)
215   FORMAT(' [illegible]BALL TO POSITION THE CURSOR OVER THE')
      [illegible]
      W[illegible]E (15,2[illegible])
220   FORMAT (' [illegible]IRED NODE AND PRESS THE ''ACCEPT'' KEY')
      GO TO 999

C  RANGE AND BEARING MESSAGES
300   CALL GSCH (191,6)
      WRITE (15,1310)
1310  FORMAT (10X,'-- RANGE AND BEARING --')
      CALL GSCH (191,60)
      WRITE(15,1322)
1322  FORMAT ('  SELECT REFERENCE POINT WITH CURSOR')
      CALL GSCH (191,114)
      WRITE (15,1325)
1325  FORMAT ('  PRESS ''ACCEPT'' KEY  TO FIX REFERENCE POINT')
      CALL GSCH (191,168)
      WRITE (15,1345)
1345  FORMAT ('  PRESS ''RETURN'' KEY TO EXIT . . .')
      GO TO 999

C  LAUNCH NODE MESSAGE
350   CALL GSCH (191,60)
```

```
          WRITE (15,1350)
    1350  FORMAT (' LOCATE ''LAUNCH'' POSITION WITH CURSOR.  PRESS')
          GO TO 475

    C  TARGET NODE MESSAGE
    400   CALL GSCH (191,60)
          WRITE (15,1400)
    1400  FORMAT (' LOCATE ''TARGET'' POSITION WITH CURSOR.  PRESS')
          GO TO 475

    C  RECOVERY NODE MESSAGE
    450   CALL GSCH (191,60)
          WRITE (15,1450)
    1450  FORMAT (' LOCATE ''RECOVERY'' POSITION WITH CURSOR.  PRESS')
    475   CALL GSCH (191,114)
          WRITE (15,1375)
    1375  FORMAT (' ''ACCEPT'' KEY WHEN CURSOR IS OVER DESIRED POSITION')
          GO TO 999

    C  RADAR SIGNIFICANT POINTS
    500   CALL GSCH (191,60)
          WRITE (15,1500)
    1500  FORMAT('MOVE CURSOR TO POSITION OF RADAR SIGNIFICANT')
          GO TO 525

    C  VISUAL SIGNIFICANT POINTS
    550   CALL GSCH (191,60)
          WRITE (15,1550)
    1550  FORMAT ('MOVE CURSOR TO POSITION OF VISUAL SIGNIFICANT')
    525   CALL GSCH (191,114)
          WRITE (15,1555)
    1555  FORMAT ('POINTS AND PRESS ''ACCEPT'' KEY.  WHEN COMPLETED')
          CALL GSCH (191,168)
          WRITE (15,1560)
    1560  FORMAT ('PRESS ''RETURN'' KEY.  ')
          GO TO 999

    C  RECALL SCENARIO MESSAGE
    600   CALL GSCH (191,6)
          WRITE (15,1600)
          CALL GSCH (191,60)
          WRITE (15,1610)
    1600  FORMAT ('SELECT MAP FROM MENU THEN ENTER VERSION')
    1610  FORMAT ('(BETWEEN 1 AND    ) FROM NUMBER PAD.')
          GO TO 999

    C  STORE SCENARIO MESSAGE
    [illegible]
          WRITE (15,1650)
          CALL GSCH (191,60)
          WRITE (15,1660)
    1650  FORMAT ('ENTER NUMBER BETWEEN 2 AND     FROM')
    1660  FORMAT ('THE NUMBER PAD TO IDENTIFY THE SCENARIO.')
          GO TO 999

    999   CALL GSCH (191)
          RETURN
          END
```

```
      SUBROUTINE MOVNOD(NODE)
      COMMON/ATT/IATT(12),IERR
      COMMON /ROUTE/ IPATH(25)
      COMMON /FLAGS/ LNODE(9),RXPT(9),LSWCH(3),IFLOW(5)
      CALL MESSAG(1)

C  SUBROUTINE MOVNOD ALLOWS THE USER TO MOVE THE POSITION OF A
C  NODE ALONG THE STRIKE PATH.  RETURN KEY #7 WILL REPLACE THE NODE
C  TO ITS ORIGINAL POSITION.
C  NODE - THE NODE THAT IS TO CHANGE.

C  GET THE CURRENT POSITION OF THE TRACKBALL
      CALL GTRACK(IX,IY)
      CALL GTRACK(IXOLD,IYOLD)
      ISWCH = .TRUE.
      I = 6*NODE-2
      CALL GENT(300)

C  GET ELEMENT NUMBER OF THE NODE
      IEL = 1+6*NODE
      CALL LAMPS(0,0,16,130)
      CALL GHLT
      CALL GCNT(1)
    5 IATT(1)=0
      CALL GSTP(0,0)
   10 CALL CHINT(KEY)
      IF (KEY .EQ. -1) GO TO 10

      IF (KEY .EQ. 1) GO TO 20
      IF (KEY .EQ. 7) GO TO 25
      IF(KEY.EQ.12)CALL DECLUT
      IF (KEY .NE. 40) GO TO 5
C
C  CYCLE TIMER INTERRUPT!  DISPLAY NEW NODE POSITION.
      CALL GTRACK(IX,IY)
      [illegible],IY,5)
      [illegible]
      CALL [illegible],IX,IY)
      IF [illegible] CALL [illegible]TLON(IX,IY)
      IF [illegible] CALL RANGE (IX,IY)
      ISWCH = .NOT. ISWCH
      GO TO 10
C
C  KEY INTERRUPT!
   20 IPATH(I)=IX
      IPATH(I+1)=IY

C  SET OFF CYCLE TIMER AND PREPARE TO EXIT FROM SUBROUTINE
25    CALL GHLT
      CALL GCNT(0)
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      RETURN
      END
0 ERRORS [illegible] COMPILED
```

```
PAGE    1        [illegible]    [illegible]    FTN [illegible]        [illegible]

    1        [illegible]
    2        [illegible]
    3        [illegible]
    4        [illegible]
    5        RETURN
    6        END
  0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE OUTSEN (IVERS)
      COMMON /MASTER/ MDF(2400)
      COMMON /ADMIN/ INFORM(120)
      INTEGER SCENE(3,20),IFCB(13)
      DATA SCENE(1, 1),SCENE(2, 1),SCENE(3, 1) /2HSC,2HEN,2H12/
      DATA SCENE(1, 2),SCENE(2, 2),SCENE(3, 2) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1, 3),SCENE(2, 3),SCENE(3, 3) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1, 4),SCENE(2, 4),SCENE(3, 4) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1, 5),SCENE(2, 5),SCENE(3, 5) /2HSC,2HEN,2H13/
      DATA SCENE(1, 6),SCENE(2, 6),SCENE(3, 6) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1, 7),SCENE(2, 7),SCENE(3, 7) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1, 8),SCENE(2, 8),SCENE(3, 8) /2HSC,2HEN,2H13/
      DATA SCENE(1, 9),SCENE(2, 9),SCENE(3, 9) /2HSC,2HEN,2H14/
      DATA SCENE(1,10),SCENE(2,10),SCENE(3,10) /2HSC,2HEN,2H[illegible]/
      DATA SCENE(1,11),SCENE(2,11),SCENE(3,11) /2H[illegible]/
      [illegible]
      DATA SCENE(1,13),SCENE(2,13),SCENE(3,13) /[illegible]/
      DATA SCENE(1,14),SCENE(2,14),SCENE(3,14) /[illegible]/
      DATA SCENE(1,15),SCENE(2,15),SCENE(3,15) /[illegible]/
      DATA SCENE(1,16),SCENE(2,16),SCENE(3,16) /[illegible]/
      DATA SCENE(1,17),SCENE(2,17),SCENE(3,17) /[illegible]/
      DATA SCENE(1,18),SCENE(2,18),SCENE(3,18) /[illegible]/
      DATA SCENE(1,19),SCENE(2,19),SCENE(3,19) /[illegible]/
      DATA SCENE(1,20),SCENE(2,20),SCENE(3,20) /[illegible]/

      IFCB(2) = 0
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      CALL [illegible]
      WRITE (13) (MDF(J),J=1,2400)
      CALL [illegible]
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE MAPP01
      CALL MAP1
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
SUBROUTINE [illegible]
CALL [illegible]
RETURN
END
```

0 ERRORS COMPILATION COMPLETE

PAGE 1 [illegible] [illegible] FTN IV [illegible] [illegible]

```
     1        SUBROUTINE POL[illegible]
     2        CALL [illegible]
     3        RETURN
     4        END
   0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE RANGE (IX,IY)
C  SUBROUTINE RANGE CALCULATES AND DISPLAYS THE RANGE AND
C  BEARING FROM THE CURSOR TO A REFERENCE POINT.  THE FIRST
C  PASS THROUGH RANGE SELECTS A REFERENCE POINT.  ALL SUBSEQUENT
C  CALLS CALCULATE AND DISPLAY THE RANGE AND BEARING.  THE
C  RANGE IS CALCULATED IN NAUTICAL MILES AND BEARING IS MEASURED
C  FROM NORTH FOR 360 DEGREES.

C          PARAMETERS

C   FIRST PASS  -
C  IX - NEGATIVE NUMBER LESS THAN -10 TO IDENTIFY FIRST CALL
C  IY - IGNORED
C   THE CURSOR COORDINATES ARE PASSED BACK TO THE CALLING ROUTINE.

C  AFTER FIRST CALL -
C  IX - X POSITION OF CURSOR
C  IY - Y POSITION OF CURSOR
C  NOTHING IS ALTERED IN THE SUBROUTINE.

      COMMON /MASTER/ MDF(2400)
      COMMON /ATT/ IATT(12),IERR
      COMMON /FLAGS/ LMODE(9),RXPT(9),LSWCH(3),IFLOW(5)
      IF (IX .GT. -10) GO TO 500

C  FIRST PASS SECTION
C  PRELIMINARIES
      CALL MESSAG(3)
      CALL LAMPS(0,0,0,130)
      CALL [illegible]
      CALL GT[illegible] (IXOLD,IYOLD)
      CALL UNLINK

C  CHECK ON INTERRUPTS
5     IATT(1) = 0
      CALL [illegible] (0,0)
10    CALL [illegible]
         IF (KEY .EQ. -1) GO TO 10
         IF (KEY .EQ. 1) GO TO 100
         IF (KEY .EQ. 7) GO TO 300
         IF (KEY .NE. 40) GO TO 5

C  CYCLE TIMER INTERRUPT
      CALL GTPACK (IX,IY)
      CALL TOLLOS (IXOLD,IYOLD,IX,IY,5)
      CALL GENT (10)
      CALL GPUT (1,100,IX,0)
      CALL GPUT (2,110,IY,0)
      CALL LATLON (IX,IY)
      GO TO 10

C  ACCEPT KEY INTERRUPT
C  REFERENCE POINT HAS BEEN IDENTIFIED
100   CALL GTPACK (IX,IY)
      RX = FLOAT(IX)
      RY = FLOAT (IY)
      CALL GENT (150)
      CALL GPUT (1,100,IX,0)
```

```
       CALL GPUT (2,110,IY,0)
       LSWCH(1) = .TRUE.
       RETURN

 300   CALL TOPMSG(2)
       CALL GEOF (120)
       RETURN

 C  SUBSEQUENT PASSES AFTER THE FIRST

 C  RUBBER BAND LINE BETWEEN 2 POINTS
 500   CALL GENT (120)
       CALL GPUT (7,53,IX,IY)
       CALL GEOM (120)
       IHTR = [illegible]
       [illegible]

 C  [illegible] TO SECOND POINT
       XDIFF = IX - FLOAT([illegible])
       YDIFF = IY - FLOAT([illegible])
       DIST = [illegible]XDIFF+YDIFF*YDIFF)

 C  CHECK ON [illegible]
       IF ([illegible] .GT. 1.0) GO TO 550
       IANGLE = [illegible]
       IF ([illegible] .LT. 0.0) IANGLE = IANGLE+180
       GO TO [illegible]
 550   IANGLE = [illegible]
       IF ([illegible] .LT. 0.0) IANGLE = IANGLE+[illegible]

 C  [illegible]
 5[illegible]  CALL [illegible]
       CALL GPUT ([illegible]DIST+.5),5)
       CALL GPUT ([illegible],5)
       RETURN
       END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE [illegible]
C     THIS [illegible]
C     [illegible]
C     [illegible]
C     [illegible]

      DIMENSION [illegible]
      COMMON [illegible]
      DATA [illegible] /2HND,2HNT,2HNS/
      CALL [illegible]
      [illegible] ([illegible],J=1,120)
      [illegible]
      RETURN
      END
0 ERRORS [illegible]
```

```
      SUBROUTINE RECALL
C  SUBROUTINE RECALL ALLOWS THE USER TO RECALL A MAP FROM DISK
C  VIA A MENU AND SELECT A VERSION WITH THE NUMBER PAD.  IT
C  WILL DRAW A PATH IF ONE EXISTS AND DISPLAY SIGNIFICANT POINTS

      COMMON /ADMIN/ INFORM(120)
      COMMON /MASTER/ MDF(2400)
      COMMON /ATT/ IATT(12),IERR
      COMMON /DISP/ IDFL(4000),MARKR(12)
      COMMON /ROUTE/ IPATH(65)
      COMMON /FLAGS/ LMODE(9),PXPT(9),LSWCH(3),IFLOW(5)

      CALL GINI(IDFL,4000,IATT,IERR)
      CALL SETUP
      CALL TOPMSG(10)
      CALL MESSAG(9)
      LMODE(1)=1

C  DISPLAY MAP OPTIONS
C  CHECK IF ONLY ONE MAP EXISTS, IF SO NO SELECTION TAKES PLACE
      NMOPT = INFORM(1)
      IF (NMOPT .NE. 1) GO TO 25
      MAP = 1
      GO TO 30

25    CALL OPTMNU(6)
      CALL SELECT (NMOPT,MAP)
      IF (MAP .EQ. 0) GO TO 35
      CALL RESET
30    CALL INSEN (MAP,0)
      CALL DEPSCN
      INFORM(3) = MAP
      IPOS = 25*MAP-4
      IF (INFORM(IPOS) .LE. 1) GO TO 50

C  USE NUMBER PAD TO INPUT VERSION
      CALL [illegible]
      CALL [illegible],INFORM(IPOS),2)
      CALL [illegible]
      CALL [illegible]
      CALL SELNUM (INFORM(IPOS),110+MAP,24,IVERS)
      IF (IVERS .LE. 0) IVERS = 1
      CALL INSEN (MAP,IVERS)

C  PUT PATH INTO IPATH ARRAY FROM MDF
      IPTR = MDF(3)-1
      DO 20 J=1,65
         IPATH(J)=MDF(IPTR+J)
20    CONTINUE
      IF (IPATH(1) .EQ. 0) GO TO 45

C  DRAW PATH AND SYMBOLS OVER LAUNCH, TARGET AND RECOVERY
35    IF (MAP .EQ. 0) CALL DEPSCN
      CALL DRPATH
      CALL GENT (310)
      CALL GPUT (1,109,IPATH(4),0)
      CALL GPUT (2,110,IPATH(5),0)
      IPOS = IPATH(2)*6-2
```

```
      CALL [illegible]
      CALL [illegible],0)
      CALL [illegible],0))
      [illegible]
      CALL [illegible]
      CALL [illegible],0)
      CALL [illegible],0)
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]
      CALL [illegible]

45    CALL [illegible]
      [illegible] = [illegible]

50    CALL [illegible](1)
      CALL [illegible]
      CALL [illegible]
      CALL [illegible] (100)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE INIT
      COMMON [illegible]
      COMMON [illegible]

C     THIS ROUTINE INITIALIZES CERTAIN COMMON PARAMETERS
      DO 110 I = 1,5
        [illegible] = 0
  110 CONTINUE

      [illegible] = [illegible]
      DO 120 I = 0,3
          [illegible] = 0
  120 CONTINUE

      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE SELECT(OPTION,CHOICE)
      IMPLICIT INTEGER (A-C,E-Z)
      DATA IRED,IYLLOW,IBLUE,IGREEN/0,1,2,3/
C ****  ROUTINE ENABLES USER TO SELECT ONE OF A LIST OF OPTIONS.
C ****  OPTIONS ARE DISPLAYED IN RED ON THE LEFT SIDE OF THE SCREEN.
C ****  USER MANIPULATES A POINTING CURSOR THAT PRECEEDS THE GREEN
C ****  CHARACTERS OF THE POINTED-AT OPTION.

C ****      OPTION          NUMBER OF MEMBERS IN LIST
C ****      CHOICE          RELATIVE POSITION IN LIST OF CHOSEN OPTION


C     TURN ON MENU AND LIGHT LAMPS
      CALL MENUON
      CALL LAMPS(0,0,0,131)

C ****  POINTER 'INC' IS 0 AND POINTS TO THE FIRST OPTION IN THE LIST
      INC=0

C ****  POSITION THE CURSOR POINTER
      CALL GENT(11)
      CALL GPUT(1,100,-339,0)
      CALL GPUT(2,110,955,0)
      CALL GEON(11)

C ****  COLOR THE FIRST ITEM IN THE LIST GREEN
      CALL GENT(111)
      CALL COLOR(IGREEN)

C ****  POLL THE FUNC. KEYBOARD FOR AN ACCEPT/REJECT
1     CALL CKINT(KEY)

      IF(KEY.NE.0)GOTO2

C     REJECT KEY HIT
      INC=INC+1
      IF(INC.[illegible]OPTION)INC=0
      Y=[illegible]

C     COLOR LAST ITEM RED AGAIN AND REPOSITION CURSOR
      CALL COLOR(IRED)
      CALL GENT(11)
      CALL GPUT(2,110,Y,0)

C ****  COLOR NEW ITEM GREEN
      CALL GENT(111+INC)
      CALL COLOR(IGREEN)
      CALL DELAY(30)
      GO TO 1

2     IF(KEY.NE.1.AND.KEY.NE.7)GO TO 1

C     ACCEPT KEY HIT
      [illegible]

C     [illegible] KEY HIT
      IF(KEY.EQ.7)CHOICE=0
```

```
      C [illegible] CHOSEN [illegible] AGAIN
            CALL [illegible]

            CALL [illegible]
            CALL [illegible]
            CALL [illegible]
            CALL [illegible],0,0,0)
            RETURN
            END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE SELNOD(NODE)
C   SELECT NODE SUBROUTINE
C   THIS SUBROUTINE ALLOWS THE USER TO MOVE A BLINKING FIGURE (CURSOR)
C   AROUND THE DISPLAY TO SELECT A NODE.  WHEN THE CURSOR GETS WITHIN
C   10 UNITS OF A NODE THE NODE WILL BLINK.  AS THE CURSOR MOVES AWAY, THE
C   NODE STOPS BLINKING. WHEN A NODE IS CHOSEN, THE NODE NUMBER IS RETURNED.

      COMMON /ATT/ IATT(12),IERR
      COMMON /ROUTE/ IPATH(65)
      COMMON /FLAGS/ LNODE(9),RXPT(9),LSWCH(3),IFLOW(5)
      CALL TOFMSG (4)
      CALL MESSAG (2)
      ISWCH = 0
C   SET OFF ALL BUT THE ACCEPT LAMP
      CALL LAMPS(0,8,16,128)

C   NODCHK CONTAINS THE NODE NUMBER OF A BLINKING NODE
      NODCHK = 0
      LSWCH(1)=.FALSE.
      IEND = IPATH(1)

C   MOVE CURSOR TO CURRENT NODE
      CALL GTRACK(IX,IY)
      CALL GTRACK (IXOLD,IYOLD)
      CALL GENT(10)
      CALL GPUT (1,100,IX,0)
      CALL GPUT (2,110,IY,0)
      CALL GEON(10)

C   SET UP CYCLE TIMER AND CHECK INTERRUPTS
      CALL GRLT
      CALL GCYT(1)
    5 IATT (1) = 0
      CALL GETT(0,0)
   10 CALL GMNT(KEY)
      IF (KEY .EQ. -1) GO TO 10

      IF (KEY .EQ. 19) GO TO 45
      IF (KEY .EQ. 1) GO TO 50
      IF(KEY.EQ.12)CALL DELUT
      IF (KEY .NE. 7) GO TO 15
      NODE = 0
      CALL GEOF (180)
      CALL TOFMSG (2)
      GO TO 55
   15 IF (KEY .NE. 40) GO TO 5

C   CYCLE TIMER INTERRUPT!
C   MOVE CURSOR AND CHECK TO SEE IF ITS CLOSE TO A NODE
      CALL GTRACK (IX,IY)
      CALL TOCLOS (IXOLD,IYOLD,IX,IY,5)
      ISWCH = 1-ISWCH
      CALL GENT(10)
      CALL GPUT (1,100,IX,0)
      CALL GPUT (2,110,IY,0)
      IF (ISWCH .EQ. 0) GO TO 16
      CALL LATLON (IX,IY)
      GO TO 18
```

```
C   RANGE AND BEARING UPDATE
16      IF (.NOT. LSWCH(1)) GO TO 18
        CALL RANGE (IX,IY)
        CALL GLMP (1,7,1)

18      DO 20 NODE = 1,IEND
            K = 6*NODE-2
           IF (IABS(IPATH(K)-IX)-10) 19,19,20
     19    IF (IABS(IPATH(K+1)-IY) .LE. 10) GO TO 22
     20 CONTINUE

C   NOT CLOSE TO A NODE, SET NODE TO 0 AND CHECK TO SEE IF NODE IS BLINKING
        NODE = 0
        IF (NODCHK .EQ. 0) GO TO 10
C   SET OFF BLINKING NODE
        CALL GLMP (1,1,0)
        IC = 0
        GO TO 40
C   SET ON BLINKING NODE
     22 IC = 1
        CALL GENT(300)
        IF (NODCHK .NE. 0 .AND. NODCHK .NE. NODE) CALL GPUT (IEL,130,3,0)
        CALL GLMP(1,1,1)
        CALL GTRACK(IX,IY)
        IEL = 6*NODE
        IF (NODE .EQ. 1) IEL=3
     40 CALL GENT(300)
        CALL GPUT (IEL,130,3,IC)
        NODCHK = NODE
        GO TO 10

C   [illegible]
45      [illegible] GO TO 16
            [illegible]
            IY = 0
            CALL RANGE (IX,IY)
        [illegible]
        [illegible] 16,120)
        IF (.NOT. LSWCH(1)) CALL GLMP (1,19,1)
        GO TO 10

C   KEY INTERRUPT!
     50 IF (NODE .EQ. 0) GO TO 5
55      CALL GHLT
        CALL GCNT(2)
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
[illegible]
```

```
      SUBROUTINE SELNUM (MAX,IENT,IEL,NUMBER)
C  SUBROUTINE SELNUM ALLOWS THE USER TO ENTER A NUMBER FROM THE
C  NUMBER PAD ON THE SPECIAL FUNCTION UNIT.  IT ALLOWS UP TO
C  FOUR DIGITS TO BE ENTERED AND DISPLAYS EACH DIGIT AS ENTERED
C  IN A SPECIFIED LOCATION.

C  PARAMETERS
C      MAX    - THE MAXIMUM ALLOWABLE NUMBER.
C      IENT   - ENTITY WHERE NUMBER IS TO BE DISPLAYED.
C      IEL    - ELEMENT IN CHARACTER AREA WHERE NUMBER ENDS (UNITS
C                 POSITION).  FOUR CHARACTER SPACES MUST BE AVAILABLE.
C      NUMBER - VALUE OF NUMBER ENTERED TO BE RETURNED TO CALLING PROG
C                 RETURNS -9999 IF NO NUMBER IS DESIRED.

      DIMENSION NO(10),INT(4)
      COMMON /ATT/ IATT(12),IERR
      DATA NO /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/

      CALL GENT (IENT)
      CALL GEON (IENT)

4     NUMBER = 0
      CALL LAMPS (7,7,7,132)

      DO 180 IDEC = 1,4
5         IATT(1) = 0
          CALL GSTT(0,0)

C  CHECK ON INTERRUPT!
10        CALL CKINT (KEY)
              IF (KEY .EQ. -1) GO TO 10
              IF (KEY .GT. 26) GO TO 10
              IF (KEY .EQ. 7) GO TO 400
              IF (KEY-1) 300,200,100
C  GET INTEGER AND DISPLAY IT
100       IF (KEY .GT. [illegible]) GO TO 110
          INT([illegible]) = [illegible]
          GO TO [illegible]

110       IF (KEY .GT. 10) GO TO 120
          IF (KEY .LT. 7) GO TO 10
          INT(IDEC) = 11-KEY
          GO TO 150

120       IF (KEY .GT. 18) GO TO 130
          IF (KEY .LT. 16) GO TO 10
          INT(IDEC) = 22-KEY
          GO TO 150

130       INT(IDEC) = 33-KEY

C  DISPLAY DIGIT RIGHT JUSTIFIED
150       DO 170 K=1,IDEC
              CALL GPUT(IEL+2-K,90,NO(INT(IDEC-K+1)+1),0)
170       CONTINUE
          CALL CLMP (1,0,1)
          CALL CLMP (1,1,1)
180   CONTINUE
```

```
C   ACCEPT KEY PRESSED,  CHECK IF NUMBER IS OVER MAXIMUM
200     [illegible]
        [illegible] = [illegible] -1
        DO 225 L=1,[illegible]
            [illegible] = [illegible]-L)
225     CONTINUE
        IF ([illegible] .LE. [illegible]) GO TO 450
        CALL [illegible]
        CALL [illegible]
        CALL [illegible] (2,7,7,[illegible])

C   [illegible] OUT DISPLAY AREA AND START OVER.
300     DO 310 L=1,4
            CALL [illegible] ([illegible]-L,90,[illegible],0)
310     CONTINUE
        GO TO [illegible]

C   [illegible]
400     [illegible] = [illegible]
450     CALL [illegible] (1,-1,0)
        RETURN
        END
0 ERRORS [illegible]
```

```
      SUBROUTINE [illegible],INIT)
      [illegible]
      [illegible] 1

C     DISPLAY [illegible] OUTPUT
      [illegible]
      DO [illegible]
         [illegible]
         [illegible]
         CALL [illegible],INIT,[illegible])
         [illegible]
         [illegible]
  120 CONTINUE
      RETURN
      END
 0 ERRORS COMPILATION [illegible]
```

```
      SUBROUTINE SETUP
C     THIS ROUTINE SETS UP THE  BORDER AND CHARACTER AREAS FOR THE MENU
      COMMON /ATT/ IATT(12),IERR
      COMMON /FLAGS/ LMODE(9),RMPT(9),LSWCH(3),IFLOW(5)
      DIMENSION NUMBER(10)
      INTEGER ENTITY
      DATA NUMBER /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA IRED,IYLLOW,IORNG,IGREEN/0,1,2,3/

C  SET UP BORDER
      CALL GBEG(2,0,0)
      CALL GPUT(3,130,2,3)
      CALL COLOR(IRED)
      CALL GPUT(5,1740,0,1)
      CALL GPUT(6,51,1033,0)
      CALL GPUT(7,52,1033,0)
      CALL GPUT(8,51,-1033,0)
      CALL GPUT(9,52,-1033,0)

C  SET UP TRACKBALL CURSOR IN ENTITY (10)
      CALL GBEG(10,0,0)
      CALL COLOR (IYLLOW)
      CALL GPUT (5,1740,0,1)
      CALL GPUT(6,1000,10,0)
      CALL GPUT (7,71,15,0)
      CALL GPUT (8,51,-30,0)
      CALL GPUT (9,70,15,15)
      CALL GPUT (10,52,-30,0)
      CALL GPUT (11,72,15,0)
      CALL GEOF(10)

C     SET UP MENU CURSOR AS ENTITY 11
      CALL GBEG(11,0,0)
      CALL GEOF(11)
      CALL GPUT(3,170,2,0)
      CALL [illegible]
      [illegible]
      [illegible]
      CALL GPUT([illegible],1[illegible],[illegible],0)

C  SET UP CHARACTER AREAS

      DO 10 I=0,10
          IENT=110+I
          IY=1000-50*I
          CALL GBEG(IENT,-330,IY)
          CALL GEOF(IENT)
          CALL COLOR(IRED)
          CALL GPUT(5,1770,1,1)
          CALL GCHA(IENT,6,0,1,25)
 10   CONTINUE

      CALL GENT(110)
      CALL GPUT(3,130,2,1)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,1)
      CALL GPUT(32,100,[illegible],0)
      CALL GPUT(33,110,1000,0)
```

```
      CALL GPUT(34,114,50,0)
      CALL GCHA(110,35,0,2,25)

C  SET UP LEG SPEED/ALT TABLE IN ENTITIES 150-160
      DO 30 I=1,11
      DO 30 K=1,12
         IENT=140+K
         CALL GSEG(IENT,1023,1000-30*K)
         CALL GOFF(IENT)
         CALL GPUT (6,104,20,0)
         CALL GCHA(IENT,7,0,1,21)
 30   CONTINUE

      CALL GCCH (150,7)
      WRITE (15,1050)
1050  FORMAT ('LEG   SPEED  ALTITUDE')

C  SET UP LINE TO CONNECT TWO POINTS FOR RANGE AND BEARING IN 170
      CALL GSEG (170,0,0)
      CALL LINEWP (7,100)
      CALL GPUT (6,100,0,0)
      CALL GPUT (7,73,0,0)

C  SET UP TOP OF SCREEN MESSAGE AREA IN 190
      CALL GSEG (190,0,1023)
      CALL GOFF (190)
      CALL GPUT (3,100,2,1)
      CALL COLOR (YELLOW)
      CALL GPUT (6,114,20,0)
      CALL GCHA (190,7,0,2,43)
      CALL  GPUT (56,104,-1000,0)
      CALL GPUT (57,114,50,0)
      CALL GCHA (190,53,0,2,43)
      CALL GPUT (107,104,-1000,0)
      CALL GPUT(107,114,50,0)
      [illegible]

C  SET UP [illegible] ENTITY 191
      CALL GSEG (191,0,[illegible])
      CALL GOFF (191)
      CALL GPUT (3,100,2,1)
      CALL COLOR ([illegible])
      CALL GCHA (191,6,0,2,50)
      CALL GPUT (57,100,0,0)
      CALL GPUT (58,110,-100,0)
      CALL GPUT (59,130,3,0)
      CALL GCHA (191,60,0,2,50)
      CALL GPUT (111,130,3,0)
      CALL GPUT (112,100,0,0)
      CALL GPUT (113,110,-150,0)
      CALL GCHA (191,114,0,2,50)
      CALL GPUT (165,100,0,0)
      CALL GPUT (166,110,-200,0)
      CALL GPUT (167,140,5,0)
      CALL GPUT (167,140,6,1)
      CALL GCHA (191,167,0,2,50)

C  SET UP LONGITUDE AND LATITUDE DISPLAY IN ENTITY 193
```

```
      CALL GREG (193,1003,140)
      CALL GEOF (193)
      CALL GPUT (6,104,34,0)
      CALL GCHA (193,7,0,1,8)
      WRITE (15,100)
100   FORMAT ('LATITUDE')
      CALL GPUT (16,104,-113,0)
      CALL GPUT (17,114,-25,0)
      CALL GCHA (193,15,0,1,4)
      CALL GPUT (23,90,1HE,0)
      CALL GCHA (193,24,0,1,4)
      CALL GPUT (29,9?,1HN,0)
      CALL GPUT (30,104,-140,0)
      CALL GPUT (31,114,-25,0)
      CALL GCHA (193,32,0,1,9)
      WRITE (15,120)
120   FORMAT ('LONGITUDE')
      CALL GPUT (48,104,-126,0)
      CALL GPUT (43,114,-25,0)
      CALL GCHA (193,44,0,1,4)
      CALL GPUT (49,90,1HD,0)
      CALL GCHA (193,50,0,1,4)
      CALL GPUT (55,90,1HM,0)

C  SET UP PATH ENTITY IN 300
      CALL GREG (300,0,0)
      CALL GEOF (300)
      LMODE(2)=1
      CALL IPWPTH
      LMODE(2)=0

C  FIGURE TO CIRCUMSCRIBE LAUNCH NODE IN ENTITY 310.
      CALL GREG (310,0,0)
      CALL GEOF (310)
      CALL COLOR (IRED)
      CALL GPUT(6,[illegible],-15)
      [illegible]
      [illegible]
      CALL GPUT(9,[illegible],-15)
      CALL GPUT(10,[illegible],-15)
      CALL GPUT(11,53,-[illegible],0)

C  FIGURE TO CIRCUMSCRIBE TARGET NODE IN ENTITY 311.
      CALL GREG (311,0,0)
      CALL GEOF (311)
      CALL COLOR (IRED)
      CALL GPUT(6,73,30,0)
      CALL GPUT(7,53,-30,0)
      CALL GPUT(8,73,15,-60)
      CALL GPUT(9,53,30,-50)
      CALL GPUT(10,73,-30,0)
      CALL GPUT(11,53,30,50)

C  FIGURE TO CIRCUMSCRIBE RECOVERY NODE IN ENTITY 312.
      CALL GREG (312,0,0)
      CALL GEOF (312)
      CALL COLOR (IRED)
      CALL GPUT(6,73,30,-15)
```

```
      CALL GPUT(7,53,0,30)
      CALL GPUT(8,53,-40,0)
      CALL GPUT(9,53,-30,-15)
      CALL GPUT(10,53,30,-15)
      CALL GPUT(11,53,40,0)

C  SET UP RADAR SIGNIFICANT POINTS IN 370 - 379
      DO 375 K=1,10
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
  375 [illegible]

C  SET UP [illegible] POINTS IN [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
  385 [illegible]

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
```

[illegible]

```
      SUBROUTINE [illegible](ISWCH)
      COMMON /MASTER/ MD(2100)

C         ISWCH=0        TURN SPS OFF
C         ISWCH=1        TURN SPS ON

      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
  100 [illegible]

  200 [illegible]
      [illegible]
      [illegible]
      [illegible]
  300 [illegible]
      [illegible]

  500 [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
[illegible]
```

```
      SUBROUTINE STORE
C  THIS SUBROUTINE ALLOWS THE USER TO SELECT WHICH VERSION ON
C  DISK THE SCENARIO WILL BE STORED.  IT CAN'T GO INTO VERSION 1
C  SINCE THIS IS ... ONLY.  IT ALSO CAN'T BE GREATER THAN
C  THE LAST VERSION NUMBER UP TO A MAXIMUM OF  6.  IT CAN BE
C  STORED ON TOP OF OTHER VERSIONS.  THE NUMBER PAD IS USED TO
C  IDENTIFY THE VERSION NUMBER.

      COMMON /ADMIN/ INFORM(150)
      COMMON /MASTER/ NDF(2400)
      COMMON /ROUTE/ IPATH(65)

C  PUT PATH ARRAY INTO THE MASTER DATA FILE
      IPTR = NDF(3)-1
      DO 10 J=1,65
         NDF(IPTR+J) = IPATH(J)
10    CONTINUE
      MAP = INFORM(3)
      CALL MESSAG(10)

      IELMT = 25+MAP-4
      IMAX = MIN0(INFORM(IELMT)+1,6)
      CALL GEHT (191)
      CALL CINT (33,IMAX,2)
      CALL CPTNUM(6)
      CALL GEON (110+MAP)

C  GET VERSION NUMBER AND WRITE TO DISK.  IF VERSION NUMBER
C  RETURNS ... ZERO, NO WRITING ON DISK WILL TAKE PLACE.
[illegible]
[illegible]
20    CALL [illegible]

C  RESET LAST ... IN THE ADMINISTRATOR (ADMIN COMMON ...)
      INFORM(... ) = 0
      INFORM(... ) = ...(3)
      IF (NO .EQ. ...) INFORM(IELMT) = IMAX

C  SET ONE LED SPEED/ALT TABLE
      IENT = 149
      DO 35   J=1,11
         CALL GEOF (IENT+J)
35    CONTINUE

C  TIME TO LEAVE
40    CALL GEOF (110+MAP)
      CALL RETURN
      CALL GEOF(191)
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE TWMSG (NO)
C  THIS SUBROUTINE CONTAINS THE MESSAGES FOR THE CHARACTER AREA
C  ABOVE THE SCENARIO.  THE TOP LINE IS CONSIDERED A TITLE LINE
      GO TO (100,100,150,200,200,200,200,200,200,200,200,200,200),NO
C  BLANK OUT MESSAGE AREA
100   CALL GSCH (190,7)
      WRITE (15,1000)
      CALL GSCH (190,55)
      WRITE (15,1000)
      IF (NO .EQ. 2) GO TO 999
      CALL GSCH (190,103)
      WRITE (15,1000)
1000  FORMAT ('                    ')
      GO TO 999

C  RANGE AND BEARING ROUTINE
150   CALL GSCH (190,55)
      WRITE (15,1150)
      CALL GSCH (190,7)
      WRITE (15,1155)
1150  FORMAT ('          RANGE    =            MILES')
1155  FORMAT ('          BEARING =            DEGREES')
      GO TO 999

C  TITLES
200   CALL GSCH (190,103)
      NUM = NO-3
      GO TO(210,250,300,350,400,450,500,550,600,650),NUM

210   WRITE (15,1210)
1210  FORMAT ('             ---CHANGE PATH MODE---')
      GO TO 999

250   WRITE (15,1250)
1250  FORMAT ('          ---SIGNIFICANT POINT MODE---')
      GO TO 999

300   WRITE (15,1300)
1300  FORMAT ('          ---LAUNCH/TARGET/RECOVERY---')
      GO TO 999

350   WRITE (15,1350)
1350  FORMAT ('             ---CHANGE PATH MODE---')
      GO TO 999

400   WRITE (15,1400)
1400  FORMAT ('            ---OPTIMIZATION MODE---')
      GO TO 999

450   WRITE (15,1450)
1450  FORMAT ('               ---CONTOUR MODE---')
      GO TO 999

500   WRITE (15,1500)
1500  FORMAT ('           ---RECALL SCENARIO MODE---')
      GO TO 999

550   WRITE (15,1550)
```

```
PAGE   2            [illegible]    [illegible]   [illegible]               [illegible]

  [illegible] 1000 FORMAT [illegible]          ---[illegible]--->
  [illegible]      GO TO [illegible]
  [illegible]
  [illegible] [illegible] WRITE([illegible],[illegible])
  [illegible] 10[illegible] FORMAT [illegible]      ---EVALUATE [illegible]--->
  [illegible]      GO TO [illegible]
  [illegible]
  [illegible] [illegible] WRITE([illegible],10[illegible])
  [illegible] 10[illegible] FORMAT [illegible]      ---DIRECTOR MODE--->
  [illegible]      GO TO [illegible]
  [illegible]
  [illegible] [illegible] CALL [illegible]
  [illegible]      RETURN
  [illegible]      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE [illegible] ([illegible], IYOLD, IY, I [illegible], INTST)

C     [illegible]
C     [illegible]
C     [illegible] IYOLD, [illegible]

      [illegible] IYOLD-IY) [illegible] 2
      [illegible]
      IF [illegible]
          IX [illegible]
          IY [illegible]
      GO TO [illegible]

 900      [illegible]
          [illegible]
 999  RETURN
      END
```

0 ERRORS [illegible]

```
      SUBROUTINE [illegible]
C     THIS [illegible]
      [illegible]
      CALL [illegible]
      [illegible]
      IF [illegible]

      CALL [illegible]
      IF [illegible]

      DO 20 [illegible]
         CALL [illegible],3,0)
   20 CONTINUE
      RETURN
      END
0 ERRORS [illegible]
```

[illegible]

```
1   [illegible]
2   [illegible]
3   [illegible]
4   [illegible]
5   [illegible]
6   [illegible]
[illegible]
[illegible]

@ASSIGN,ST,CP
```

DISTRIBUTION LIST

Director, Engineering Psychology
 Programs (Code 455)
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217 (5 cys)

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314 (12 cys)

CDR Paul Chatelier
Office of the Deputy Under Secretary
 of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Dr. Stuart Starr
Office of the Under Secretary of Defense
(C3I)
Pentagon, Room 3C200
Washington, D.C. 20301

Dr. Craig Fields
Director, Cybernetics Technology Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Office of the Chief of Naval
 Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Commanding Officer
ONR Eastern/Central Regional Office
ATTN: Dr. J. Lester
Bldg. 114, Section D
666 Summer Street
Boston, MA 02210

Commanding Officer
ONR Branch Office
ATTN: Dr. Charles Davis
536 South Clark Street
Chicago, IL 60605

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Commanding Officer
ONR Western Regional Office
ATTN: Mr. R. Lawson
1030 East Green Street
Pasadena, CA 91106

Tactical Development & Evaluation
 Support
Code 230
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Operations Research Program
Code 434
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Statistics and Probability Program
Code 436
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Information Systems Program
Code 437
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

CDR Kent Hull
Code 430B
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

CDR P.M. Curran
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. James McGrath
Code 311
Navy Personnel Research and
Development Center
San Diego, CA 92152

Management Support Department
Code 210
Navy Personnel Research and
Development Center
San Diego, CA 92152

Naval Electronics Systems Command
Human Factors Engineering Branch
Code 4701
Washington, D.C. 20360

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375 (6 cys)

Commander, Naval Electronics
Systems Command
Command and Control Division
Code 530
Washington, D.C. 20360

Dr. John Silva
Head, Human Factors Division
Naval Ocean Systems Center
San Diego, CA 92152

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, VA 22202

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 92940

Dr. Joseph Zeidner
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. H.W. Sinaiko
Smithsonian Institution
801 N. Pitt Street
Alexandria, VA 22314

Office of the Chief of Naval
Operations OP942
Pentagon
Washington, D.C. 20350

Office of the Chief of Naval
Operations OP987C
R&D Plans Division
Washington, D.C. 20350

Commander
Naval Electronics Systems Command
C3 Project Office
PME 10801
Washington, D.C. 20360

Dr. Julie Hopson
Human Factors Engineering Division
Code 604
Naval Air Development Center
Warminster, PA 18974

M.L. Metersky
Naval Air Development Center
Code 5424
Warminster, PA 18974

Dr. Edgar M. Johnson, Director
Ogranizations & Systems Research
Laboratory
U.S. Army Research Laboratory
5001 Eisenhower Avenue
Alexandria, VA 22333

Mr. Victor Monteleon
Naval Ocean Systems Center
San Diego, CA 91252

Commander, Naval Electronics
Systems Command
ELEX-03
Washington, D.C. 20360

CDR Richard Schlaff
NIPSSA
Hoffman Bldg. #1
2461 Eisenhower Avenue
Alexandria, VA 22331

Dr. Chantee Lewis
Management Department
Naval War College
Newport, RI 02840

Dr. John Shore
Naval Research Laboratory
Code 5403
Communications Sciences Division
Washington, D.C. 20375

Dr. Meredith Crawford
American Psychological Association
Office of Educational Affairs
1200 17th Street, N.W.
Washington, D.C. 20036

Dr. William Dejka
ACCAT
Naval Ocean Systems Center
San Diego, CA 92152

Mr. Merlin Malehorn
Office of the Chief of Naval
Operations (Op 102)
Washington, D.C. 20350

Dr. S.D. Epstein
Analytics
2500 Maryland Road
Willow Grove, PA 19090

Dr. Amos Freedy
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. G. Hurst
University of Pennsylvania
Wharton School
Philadelphia, PA 19174

Dr. Miley Merkhofer
Stanford Research Institute
Decision Analysis Group
Menlo Park, CA 94025

Mr. George Pugh
Decision Science Applications, Inc.
1500 Wilson Boulevard
Arlington, VA 22209

Dr. Arthur Siegel
Applied Psychological Services
Science Center
404 E. Lancaster Street
Wayne, PA 19087

LCDR J.A. Sears
Department of MIS
College of Business Administration
University of Arizona
Tucson, AZ 85721

I.R. Mirman
Asst for Special Projects
HQ AFSC-DL
Andrews AFB, MD 20334

Mr. Joseph Wohl
MITRE Corporation
Box 208
Bedford, MA 01730

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology
 Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Mr. Tim Gilbert
MITRE Corporation
1820 Dolly Madison Blvd.
McLean, VA 22102

Mr. Leslie Innes
Defense & Civil Institute of
 Environmental Medicine
P.O. Box 2000
Downsview, Ontario M3M 3B9
Canada

Dr. Rex Brown
Decision Science Consortium
Suite 421
7700 Leesburg Pike
Falls Church, VA 22043

Dr. A.C. Miller, III
Applied Decision Analysis
3000 Sand Hill Road
Menlo Park, CA 94025

DATE
FILMED
-8